НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та Обчислювальної техніки

Кафедра Обчислювальної техніки

«На правах рукопису»
УДК 004.418

До захисту допущено:
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
«___»_____2021_ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Комп'ютерні системи та мережі»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Спосіб побудови інформаційного навчально-методичного комплексу, та його аналіз і оцінка якості»

Виконав:
студент VI курсу, групи ІО-91мн
Буцький Юрій Петрович _____

Керівник:
Доцент кафедри ОТ, к.т.н., доцент,
Ткаченко Валентина Василівна _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н., професор,
Кулаков Юрій Олексійович _____

Рецензент:
Професор кафедри АСОІУ, д.т.н., професор,
Стеценко Інна Вячеславівна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет (інститут) Інформатики та обчислювальної техніки
(повна назва)

Кафедра Обчислювальної техніки
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою
Спеціальність 123. Комп'ютерна інженерія
(код і назва)

Спеціалізація 123. Комп'ютерні системи та мережі
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Стіренко С.Г.
(підпис) (ініціали, прізвище)
«____» _____ 2021 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Буцького Юрія Петровича
(прізвище, ім'я, по батькові)

1. Тема дисертації Спосіб побудови інформаційного навчально-методичного комплексу, та його аналіз і оцінка якості
Науковий керівник дисертації к.т.н., доцент Ткаченко Валентина Василівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «12» березня 2021 р. № 809-с

2. Строк подання студентом дисертації 26 квітня 2021 р.

3. Об'єкт дослідження навчально-методичний комплекс, який реалізовується в рамках інформаційної системи автоматизації навчальним процесом, та його структурні компоненти тестування роботи

4. Предмет дослідження спосіб розробки заданого навчально-методичного комплексу для забезпечення масштабованості інформаційної системи, а також аналіз отриманої моделі роботи системи

5. Перелік завдань, які потрібно розробити: аналіз принципів роботи інформаційних систем забезпечення навчанням; порівняння існуючих систем забезпечення навчального процесу; формування складових роботи освітнього процесу, на основі яких працюватиме автоматизований комплекс; проєктування моделі розробленого комплексу в рамках заданої інформаційної системи;

обґрунтування програмних і апаратних складових, необхідних для роботи платформи; розробка конфігурації роботи компонентів платформи; тестування стану роботи розробленої системи та її компонентів; оцінка і аналіз складових системи.

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	д.т.н., професор Кулаков Ю. О.		

7. Дата видачі завдання 26 жовтня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1.	*Затвердження теми роботи*	*01.12.2020-15.12.2020*	
2.	*Вивчення та аналіз завдання*	*15.12.2020-31.01.2021*	
3.	*Аналіз підходів до побудови інформаційних навчальних систем*	*01.02.2021-15.02.2021*	
4.	*Проєктування моделі навчально-методичного комплексу*	*15.02.2021-31.02.2021*	
5.	*Програмна реалізація комплексу і залежних компонентів системи*	*01.03.2021-10.04.2021*	
6.	*Оформлення пояснювальної записки*	*11.04.2021-25.04.2021*	
7.	*Передзахист*	*26.04.2021*	
8.	*Захист*	*18.05.2021*	

Студент _____ Ю.П. Буцький
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ В.В. Ткаченко
(підпис) (ініціали, прізвище)

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Спосіб побудови інформаційного навчально-методичного комплексу, та його аналіз і оцінка якості

студентом: Буцьким Юрієм Петровичем

Дана магістерська дисертація складається із вступу та 4 розділів. Сумарний об'єм роботи: 125 аркушів тексту, містить 41 ілюстрацію та 2 таблиці. В ході роботи було використано літературу з 42 різноманітних джерел.

Актуальність. У світі в різних компаніях і організаціях все більше приходить ідея розробок інформаційних систем для потреб різних організацій або бізнес проєктів. Із розвитком технологій все більше сфер діяльності людства мають змогу впровадження комп'ютерних засобів для прискорення виконання різних процесів, збереження окремих даних, або виконання різного додаткового функціоналу. Розвиток інформаційних систем для потреб освіти є сучасною тенденцію покращення організації навчального процесу у різних навчальних закладах.

Вже на поточний день існує ряд окремих розроблених систем, які реалізовують частковий функціонал впровадження освітнього процесу в такі інформаційні системи. Та проте, не існує досі чіткої структури навчально-методичної складової, яка може забезпечити діяльність інформаційної системи у правильному режимі. Крім того, існуючі системи не мають ряд окремих додаткових сервісів, які надаються сучасними засобами програмування, перевірку стану роботи окремих компонентів, та робити оцінку якості роботи програми.

.**Мета і завдання дослідження.** Мета магістерської дисертації полягає в розробці окремої методології побудови навчально-методичного комплексу, який може працювати гармонійно в рамках деякої інформаційної системи, а також відповідати своїми компонентами складовим навчального процесу. Крім того, заданий комплекс мусить мати окремі складові, які забезпечують моніторинг

роботи компонентів платформи, а також тестування якості написання програмного коду.

Для досягнення поставленої мети, було поставлено і вирішено наступні завдання:

- Аналіз принципів роботи інформаційних систем забезпечення навчанням;
- Порівняння існуючих систем забезпечення навчального процесу;
- Формування складових роботи освітнього процесу, на основі яких працюватиме автоматизований комплекс;
- Проєктування моделі розробленого комплексу в рамках заданої інформаційної системи;
- Обґрунтування програмних і апаратних складових, необхідних для роботи платформи;
- Розробка конфігурації роботи компонентів платформи;
- Тестування роботи розробленої системи та її компонентів;
- Оцінка і аналіз складових системи.

Об’єкт дослідження – навчально-методичний комплекс, який реалізовується в рамках інформаційної системи автоматизації навчальним процесом, та його структурні компоненти тестування роботи.

Предмет дослідження – спосіб розробки заданого навчально-методичного комплексу для забезпечення масштабованості інформаційної системи, а також аналіз отриманої моделі роботи системи.

Методи досліджень базуються на теоретичному підході розгляду теорії інформаційних систем, а також на практичній основі реалізації сервісів тестування і оцінки якості роботи отриманих компонентів системи Теоретичний підхід базується на аналізі компонентів освітнього процесу, та порівняльній характеристики з існуючими системи автоматизації навчання. На основі цих методів було застосовано синтез моделі навчально-методичного комплексу, та обґрунтовано складові необхідні для реалізації додатку. Практична частина побудована на основі обраних складових, і окремих засобів моніторингу і

тестування роботи програмного коду, для можливості досягнення оцінки роботи системи.

Наукова новизна одержаних результатів. Розроблений спосіб побудови навчально-методичного комплексу формує власну структуру складових організації освітнього процесу, яка може бути легко адаптована до різних інформаційних систем. Крім того, отримана модель повністю відповідає властивостям масштабованої складної інформаційної системи, та легко адаптується до сучасної мікросервісної архітектури програмних додатків.

Практична значимість результатів дослідження. Отриманий в результаті дослідження навчально-методичний комплекс використовується в рамках існуючої платформи автоматизації навчального процесу KPI-Connect, який впроваджено у використанні на кафедрі Обчислювальної техніки Факультету Інформатики та Обчислювальної техніки в НТУУ «Київський Політехнічний Інститут ім. Ігоря Сікорського». За рахунок наявності реалізованих систем моніторингу і тестування програмного коду, існує можливість у підтримці роботи в системи у напівавтоматизованому режимі, в залежності від отриманих повідомлень і попереджень платформи, і в деякому плані є практичною новизною у використанні в інформаційних системах організації навчального процесу.

Публікації. Основні результати дисертації було апробовано і .опубліковано в 2 наукових працях, серед яких всі – матеріали наукових конференцій:

1. Butskyi Y. Integration of the anti-plagiarism module for information educational platforms [Електронний ресурс]. / Y. Butskyi. // Science and Technology of the XXI Century: Proceedings of the XXI International Students R&D Online Conference (Kyiv, 17 December, 2020). – Kyiv, 2020. – Part III. – pp. 20-21. – режим доступу: https://kamts2.kpi.ua/sites/default/files/Part_III_Section_9_1.pdf

2. Butskyi Y. Development prospects of universities' educational and methodological information systems on the example of the "KPI-Connect" platform [Електронний ресурс]. / Y. P. Butsyi, K. M. Hryshchenko, I. A. Klymenko. //. I International Scientific and Theoretical Conference "Formation of Innovative Potential

of World Science” (Tel Aviv, State of Israel, 7 May, 2021). – Tel Aviv, 2021. – Vol. 1. – pp. 163-170. – режим доступу: https://doi.org/10.36074/scientia-07.05.2021

Ключові слова

інформаційна система, навчально-методичний комплекс, масштабованість, мікросервісна архітектура, моніторинг комп’ютерних систем, оцінка якості програмного продукту, Docker, Prometheus, Grafana, SonarQube.

ЗМІСТ

ВСТУП

На поточний час, нові технології з галузі комп’ютерної техніки з’являються чи не кожен день. Із доступом і розвитком мережі Інтернет, людство має можливість використовувати новітні технології чи не в будь-якій поточній галузі. На даний момент, дуже важливо мати змогу впровадити новітні комп’ютерні засоби в різноманітні традиційні галузі, в тому числі у систему навчання.

Серед існуючих способів інтеграції комп’ютерних технологій у навчання можна назвати реалізацію спеціалізованих інформаційних систем, що можуть забезпечити можливість автоматизувати ряд необхідних процесів, які відбуваються у навчальних закладах. Побудова такого виду системи потребує врахування ряду різних факторів, які є відносно специфічними при необхідності реалізації комп’ютерних технологій у ряді інших галузей чи бізнес проєктів.

Дана робота присвячена аналізу способів побудови різних інформаційних систем, пов’язаних із навчальним процесом, в результаті якого буде сформований свій метод реалізації такої моделі даних. В процесі виконання роботи будуть розглянуті ряд необхідних факторів потрібних для правильної організації комп’ютерної системи автоматизації навчального процесу. Крім того, будуть визначені особливості, які відокремлюють поточну реалізацію від інших систем, розроблених для різних навчальних закладів, або виконаних як комерційний бізнес проєкт.

В результаті виконання роботи буде представлена модель реалізації навчально-методичного комплексу, його застування в рамках інформаційної платформи автоматизації навчального процесу, а також її реалізація із використанням зазначених сучасних технологій проєктування і програмування. Розробка такої платформи в рамках деякого навчального закладу неодмінно допоможе прискорити ряд необхідних навчальних процесів і спростити роботу із великими об’ємами даних. Крім того, отримана розроблена платформа буде цілком відповідати організації структури університету, що таким чином

дозволить легко її інтегрувати в поточний електронний простір навчального закладу, а також зв’язати із іншими засобами і платформами через мережу Інтернет.

На основі розробленого комплексу також будуть представлені засоби проведення моніторингу стану компонентів, і тестування на якість написання програмного коду. Використання заданих засобів наддасть змогу провести аналіз виконаного дослідження, і сформувати оцінку якості розробленого програмного додатку.

РОЗДІЛ 1
АНАЛІЗ СУТІ І МЕТИ НАВЧАЛЬНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1. Загальні положення

Дані, які оброблюються сучасними комп'ютерними пристроями і технологіями, потребують чітко визначеної деякої організації, за якими відбувається їх процес обробки. Так само, якщо розглядати деякі бізнес проєкти, які потребують використовування сучасних технологій, регламент опрацювання даних, їх роль і організація у комп'ютерній техніці мусить бути визначена якоюсь окремою ідеєю, або абстракцію. Виходячи із такої необхідності в організації даних в усіх сучасних компаніях, бізнес програмах, різних галузях, тощо виникає окреме поняття як інформаційна система.

Різні експерти мають визначення щодо терміну «інформаційна система» по різному. Серед відомих визначень цього терміну можна назвати наступні:

- Інформаційна система може бути визначена технічно як набір взаємопов'язаних компонентів, які збирають, оброблюють, зберігають і розповсюджують інформацію для підтримки прийняття рішень та контроль в організації;
- Інформаційна система є комбінацією апаратного, програмного забезпечення та телекомунікаційних мереж, які люди будують і використовують для збору, створення і розповсюдження корисних даних, як правило, в організаційних умовах;
- Інформаційні системи – це взаємопов'язані компоненти, що працюють разом для збору, обробки, збереження та розповсюдження інформації для підтримки прийняття рішень, координації, контролю, аналізу та візуалізації в організаціях.

Як можна побачити із визначень, вони в більшій мірі сфокусовані на двох шляхах опису інформаційних систем: *компоненти*, які створюють інформаційну систему, і *ролі,* яку ці компоненти виконують в організації. Розглянемо окремо кожен із шляхів опису цього терміну [19, 42].

1.1.1. Компоненти інформаційної системи

В цілому, інформаційні системи складаються з п’яти основних компонентів: комп’ютерне апаратне і програмне забезпечення, дані, люди і процеси. Також, нещодавно було визначено ще один, не менш важливий компонент, як телекомунікаційні мережі. З основних компонентів, перші три компоненти визначають як інформаційні технології, або скорочено IT. Останні два компоненти, а саме люди і процеси, не випадково існують як голові елементи інформаційних систем, адже вони дозволяють відокремити поняття інформаційних систем від більш технічних питань, якими являються інформаційні технології, наприклад, у комп’ютерних науках.

В цілому, схематично компоненти інформаційної системи можна зобразити як на рис. 1.1:

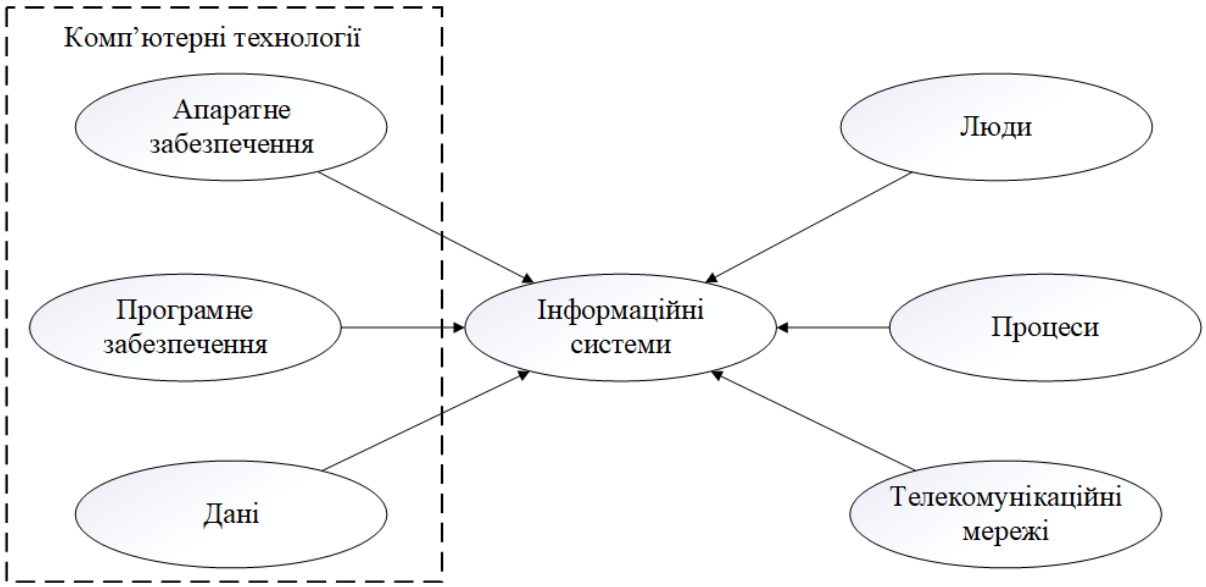


Рис. 1.1. Схема структурних складових інформаційної системи

Розглянемо кожен із компонентів самостійно:

Комп’ютерне апаратне забезпечення.

Апаратне забезпечення в інформаційних системах – фізична частинка, до якої можна доторкнутися, і яка призначена для обробки інформації на апаратному рівні в системі. Цією фізичною частиною можна вважати все, яке користувач системи використовує для отримання результатів взаємодії із системою – комп’ютер, ноутбук, клавіатура, миша, віддалений сервер, дискові і

флеш накопичувачі, тощо. Всі ці розглянуті елементи в цілому створюють загальне комп'ютерне апаратне забезпечення інформаційної системи для певної компанії.

Комп'ютерне програмне забезпечення.

Програмне забезпечення у інформаційних системах виступає такою складовою, яка виступає як набір необхідних інструкцій, який каже апаратному забезпеченню що виконувати. Особливістю програмного забезпечення є те, що до нього не можливо доторкнутися – воно не є матеріальним. В цілому, ця складова ділиться на два окремі класи: Операційні системи (системне програмне забезпечення) і прикладне програмне забезпечення.

Перший клас в цілому відповідає за проміжний інтерфейс взаємодії між апаратним і прикладним програмним забезпеченням. Операційні системи дозволяють правильно управляти апаратним забезпеченням, керувати даними і програмними файлами, і надавати користувачу можливості «розумно» керувати комп'ютером.

Прикладне програмне забезпечення призначене для деяких специфічних задач користувачів. Цільовими елементами цього класу виступають користувацькі програми і додатки, які працюють на певній призначеній для них Операційній системі.

Дані.

Дані можна представити у вигляді набору пов'язаних між собою фактів. Цими фактами можна представити будь-що, наприклад, адреса будинку певного користувача (країна, місто, вулиця, номер будинку), персональні дані в певній соціальній мережі, опис певної нещодавньої події в будь-якій частині світу – все це є де-якими частинами певних даних.

Як правило, частини не пов'язаних між собою даних не несуть багато користі для оперування в інформаційних системах. Проте, якщо дані були згруповані, відфільтровані, і організовані у деякі бази даних, то вони можуть стати головним інструментом для оперування бізнесу в рамках певної інформаційної системи. Аналіз і оперування такими блоками даних допомагають

організаціям для контролю прийняття рішень, надання певного результату обробки певним користувачам системи, а також в цілому дозволяє вести контроль продуктивності бізнесу.

Якщо розглядати всі ці три окремі компоненти в цілому, то їх структурну модель можна представити як на рис. 1.2:

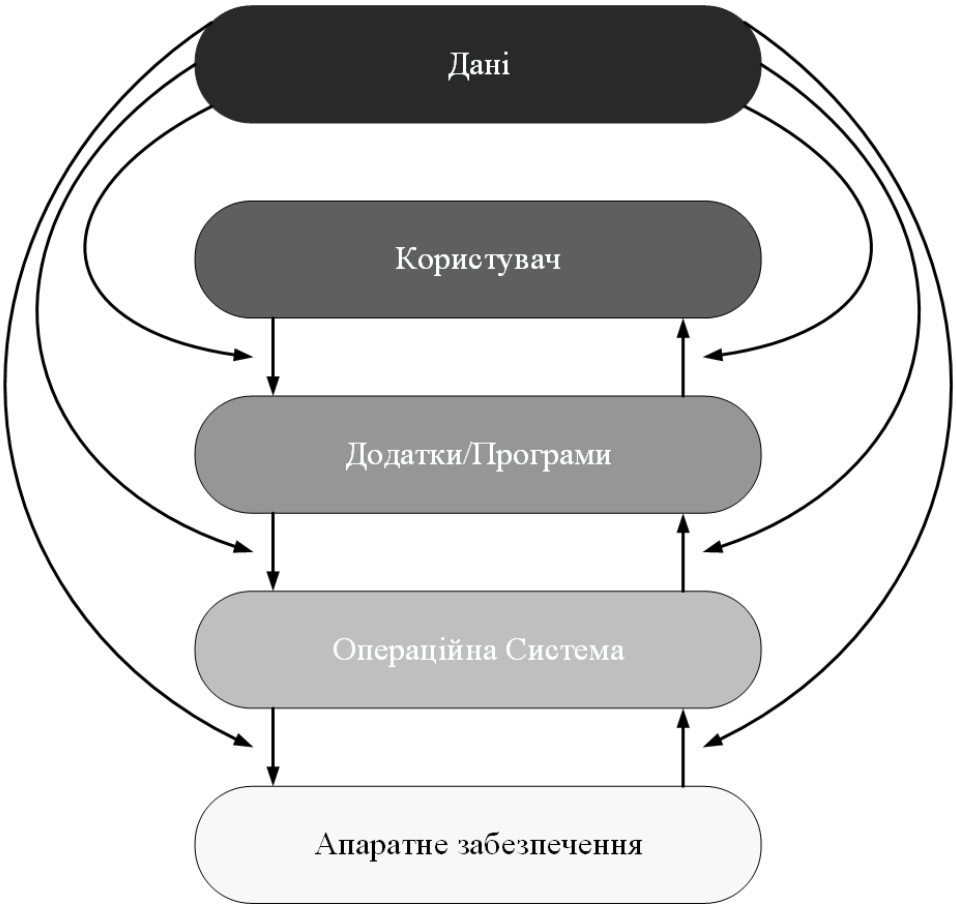


Рис. 1.2. Взаємодія користувача із трьома компонентами технологій інформаційної системи

Як можна побачити на рисунку, дані виступають елементом взаємодії на кожному рівні обробки апаратним і програмним забезпеченням, що неодмінно означає їх важливість у використані при побудові певної інформаційної системи.

Телекомунікаційні мережі.

Мережі представляють у собі засоби для з'єднання, зв'язку між окремими комп'ютерними системами, різними портативними пристроями, або в цілому кажучи, між різним апаратним забезпеченням для передачі інформації. Зв'язки можна організовувати як і за допомогою дротів, так і без їх допомоги. Наприклад, зв'язок між окремими стаціонарними комп'ютерами в окремій організації, чи

між окремими комп'ютерними мережами в рамках одного дата центру виконується в рамках дротового з'єднання. Бездротове з'єднання може використовуватися при підключенні мобільних комп'ютерних пристроїв, смартфонів, вбудованих апаратних систем, мікро або радіо хвилі.

Крім з'єднання окремі мережі групуються за певними технологіями, або протоколами підключення. Таким чином, надаючи певним користувачам необмежену можливість з будь-якої локації отримати і обробити певні дані у своїй інформаційній системі. Виходячи з цього, можна зробити висновок, що Телекомунікаційні мережі виступають поєднанням апаратного і програмного забезпечення у сьогоднішній час, для забезпечення безперервності з'єднання для обробки даних у розподілених комп'ютерних системах, якими користуються чи не кожна відома організація.

Люди.

Крім технологічних компонентів, які були розглянуті попередньо, дуже важливим елементом сучасної інформаційної системи слід вважати людей, які в цілому задіяні в них. Люди виконують роль оперування структурними елементами кожної інформаційної системи, інтеграції додаткових операцій, і в цілому для аналізу результатів обробки даних. Сюди слід враховувати всі типи людей, які необхідні для правильної підтримки роботи інформаційної системи, такі як розробники і операційні менеджери, керівники і директори відділів організацій, персонал служби підтримки користувачів, тощо.

Процеси.

Останнім, але не менш важливим компонентом виступають процеси – набір кроків, які необхідно зробити інформаційній системі, щоб отримати бажаний результат або ціль. Як правило, процеси мають обговорюватися і документуватися в цілях розробки певної інформаційної системи, щоб отримані результати обробки даних задовольняли бізнес потреби вибраної окремої організації. Тим не менш, основною ціллю розвитку сучасних процесів у різних організаціях при їх інтеграції у інформаційні системи – досягнення певного рівня

автоматизації виконання тої, чи іншої задачі за для отримання бажаного результату обробки певних даних [19].

1.1.2. Ролі інформаційних систем

Виокремлюють шість головних ролей, які інформаційні системи виконують у організаціях. Деякі з цих ролей підпадають під визначення самого терміну, зазначеного вище, але тим не менш саме вони визначають необхідність у використовуванні інформаційних систем у бізнесі:

Управління операціями.

Інформаційні системи, як правило, допомагають багатьом організаціям і бізнесам в управлінні різними операціями, як і складними, так і простими, такі що виконуються майже кожен день, або дуже рідкі, але потрібні дії, які складно або неможливо реалізувати без використання комп'ютерних технологій. Як приклад, можна розглянути такі операції, з якими може справлятися інформаційна система: відстежування місячної зарплатні співробітників, сплата податків, облік відвідування і робочого часу, обробка акаунтів користувачів, управління активами і персональним інвентарем як кожного співробітника, так і окремих масивних заводів і обладнань, розташованих у віддалених куточках земної кулі, тощо. Ведення даних щодо вказаних, звичайних операцій в організаціях за допомогою інформаційних систем значно спростило управління великим обсягом інформації, які до використання комп'ютерів було необхідно оброблювати вручну.

Підтримка взаємодії з клієнтами.

При реалізації того, чи іншого продукту в бізнесі чи організації, дуже важливо щоб користувачі або клієнти цього продукту отримали найкращий сервіс і найбільшу вигоду у роботі із конкретною компанію. Інформаційні системи можуть допомогти вирішувати такі проблеми, коли необхідно виконати якусь просту дію для клієнта, але поточне рішення потребує багато додаткових кроків, чи втручання зі збоку персоналу компанії.

Як приклади щодо цієї ролі інформаційних систем можна навести системи сканування товару у супермаркетах, чи знімання готівки у касах банків. Так, у першому випадку необхідно багато часу витрачати на чергу до каси із продавцем. Крім того, можуть виникати ситуації коли деякий товар має помилковий штрих-код сканування, і відповідальний менеджер витрачає багато часу для вирішення ще додаткової проблеми, яке може при цьому дуже дратувати кінцевих клієнтів магазинів. Щодо ситуації з банками, тут виникає така ж ситуація з чергами до каси в банк, які можуть займати в тому числі клієнти, яким необхідно не просто отримати готівку, а сплатити деякі квитанції, або зробити грошовий переказ на іншу рахунок.

В результаті існуючих недоліків щодо обслуговування клієнтів, були розроблені такі інформаційні системи, як каса самообслуговування в магазинах, або банкомати для зняття готівки в банках.

Прийняття рішень.

У будь-якій організації чи бізнесі менеджери мають приймати рішення щодо різних питань. Для прийняття таких рішень їм потрібно аналізувати велику кількість інформації щодо деяких результатів процесів компанії, на основі якісно оброблених даних. Інформаційні системи можуть допомогти в обробці зібраних даних компанії і генерації результатів аналізу у якісній формі, що може набагато спростити таким менеджерам зрозуміти позиції організації, і прийняти правильне рішення. Прикладом до цього типу ролі можна назвати систему продаж, яка буде аналізувати скільки доходу отримує ресторан в останню годину роботи, і ці дані можуть допомогти менеджерам вибрати найкраще рішення, щоб вибрати час закриття закладу раніше.

Співпраця в командах.

Крім зазначених вище ролей, деякі інформаційні системи дозволяють людям працювати разом в командах, у будь-який час та у будь-якому місті. Люди мають можливість за допомогою таких систем організовувати онлайн зустрічі і конференції, ділитися спільним доступом до документів і додатків, взаємодіяти за допомогою мікрофону, відеокамери, тощо.

Серед прикладів до зазначених інформаційних систем можна назвати сайти із соціальними мережами, а також різні системи співпраць, які дозволяють їх користувачам ділитися різною інформацією, виконувати задачі одночасно, а також шукати нові соціальні зв’язки з іншими різними групами контактів.

Отримання конкурентних переваг.

Для будь-якої компанії чи не найголовнішим ефектом прогресу є побудовою стратегії і планів для здобуття найбільших конкурентних переваг на ринку серед інших схожих фірм чи організацій. Розробка і використання передових технологій у інформаційних системах виступає фундаментальною частиною в плані стратегій розвитку визначеної компанії на ринку, яка також може надати передові позиції серед конкурентів.

Як приклад, можемо розглянути компанію Apple, яка при розробці свого смартфону iPhone вперше реалізувала технологію голосового розумного персонального асистенту Siri. Під час представлення цієї технології жодна компанія на ринку не мала щось схожого на Siri, що надало компанії один із проривних ефектів захоплення ринку смартфонів більш ніж як на 50%. На поточний момент, вже інші компанії розробили аналоги своїх голосових асистентів, як компанія Microsoft представила Cortana, і почала використовувати її у своїх продуктах, або компанія Google створила Google Assistant, який також тепер використовується у смартфонах на операційній системі Android. Але у свій час створення і початок використання асистенту Siri допомогло Apple отримати перевагу на ринку смартфонів.

Поліпшення індивідуальної продуктивності.

Інформаційні системи крім зазначених вище важливих ролей також мають відігравати можливість поліпшення продуктивності при виконанні тих чи інших цілей. Сюди можна вказати використання безліч сучасних засобів і технологій, які можуть допомогти прискорити і покращити процес виконання тих чи інших операцій.

Наприклад, використання текстових препроцесорів як програма Microsoft Word включає у собі велику кількість окремих модулів, яка може допомогти із

процесом написання і форматування різних документів, побудови діаграм і інтеграції даних з будь-яких інших сторонніх продуктів Microsoft і не тільки. Крім того, використання вбудованої системи макросів може дозволити спростити виконання де-яких базових операцій із редагування документу до мінімуму, що значно підвищить продуктивність користувачів цієї системи [41].

Як результат, всі зазначені ключові ролі використання інформаційних систем можна зобразити як на рис. 1.3:

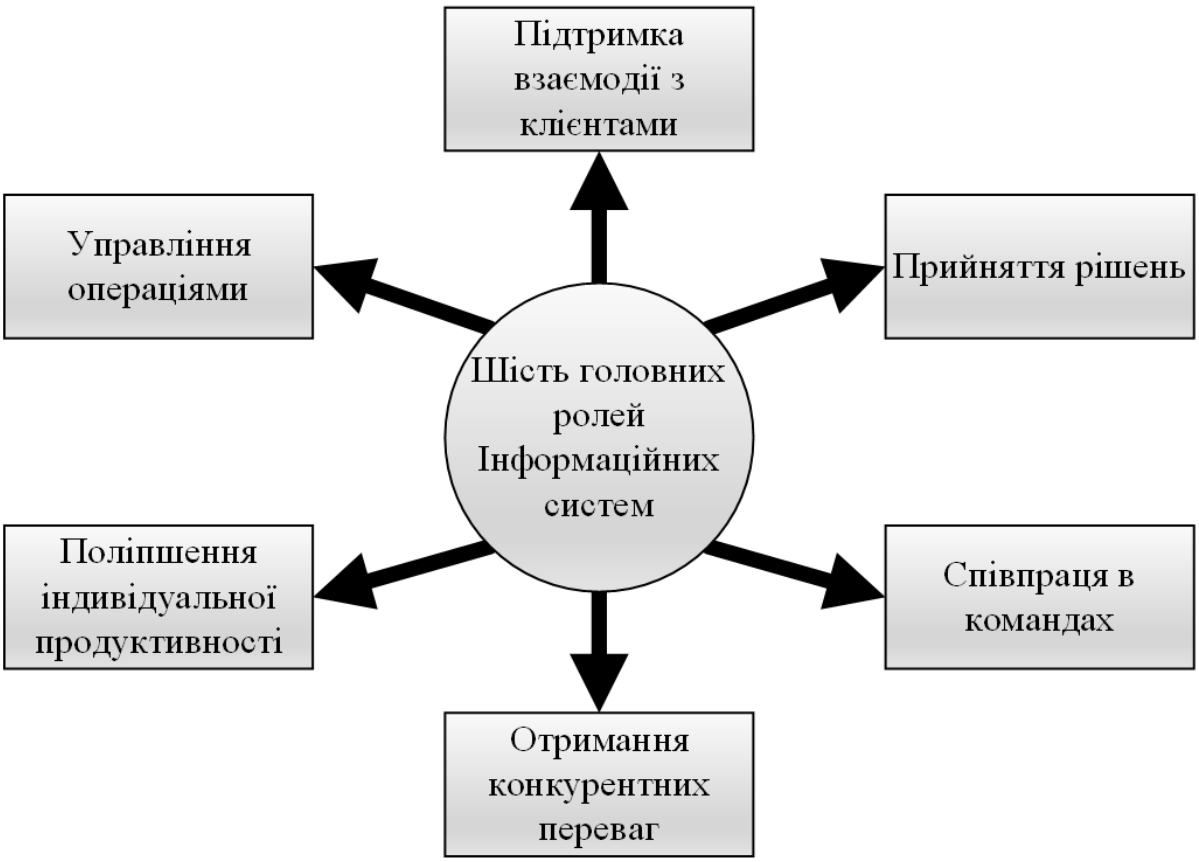


Рис. 1.3. Головні ролі використання інформаційних систем в організаціях [41]

1.2. Типи інформаційних систем

Крім розглянутих основних складових інформаційних систем, та їх ролей у організаціях, дуже часто їх ще розподіляють на різні типи. Розподіл найчастіше відбувається відповідно до ролей, які ці системи виконують, так і структурою важливості даних, які оброблюються в рамках інформаційної системи різними співробітниками компаній.

Найчастіше експерти розглядають три основні типи інформаційних систем, які використовуються у організаціях. Зазначені типи часто зображають як діаграму у вигляді піраміди, де різні типи інформаційних систем розташовані за важливістю прийняття структурованих рішень користувачами в рамках деякого бізнесу. Дана діаграма має вигляд як на рис. 1.4:

Рис. 1.4. Структура типів інформаційних систем в організаціях

Розглянемо кожен рівень організації типів інформаційних систем окремо. Слід зазначити, що деякі ролі інформаційних систем відповідають певному із рівнів типів систем із показаної діаграми, як це буде показано згодом.

Рівень операційної підтримки.

Операційний рівень пов’язаний із виконанням щоденних справ у бізнесі, і найчастіше пов’язано із процесами обробки транзакцій. Головною особливістю цього рівня полягає в тому, що користувачі таких систем виконують структуровані рішення. Це означає, що користувачам потрібно користуватися заздалегідь визначеними деякими правилами, чи базою знань, яке може їм допомогти із визначенням деякого рішення.

На даному рівні розглядаються три найпопулярніші типи систем: системи управління ланцюгами поставок (SCM), системи управління відносинами з клієнтами (CRM) і системи обробки транзакцій.

Перший тип систем на операційній підтримці SCM пов’язаний із керуванням потоків продукції, даних, грошей і інформації через весь ланцюжок поставок, який починається із постачальників сировини, продовжується через компанії обробки матеріалів і даних, і закінчується дистриб’юторами та роздрібними торгівцями. Такий тип систем контролює увесь процес передачі інформації між різними компаніями із контролю надання послуг кінцевому клієнту, коли в той же час користувачам системи непотрібно приділяти великих зусиль для управління системою, чи прийняті рішень щодо деякого отриманого результату обробки даних на попередньому кроці. Як приклад роботи такої системи можна назвати реалізацію платформи продажу продукції в супермаркеті. Коли касир друкує чек за покупку, то в інвентарі системи автоматично прибираються відповідні позиції щодо деякої продукції, і через деякий час система просто вказує яку нову продукцію необхідно замовити від постачальника, і так само автоматично відправляє запит на генерацію нової поставки.

Другий тип систем – системи управління відносин із клієнтами (CRM). Такий тип систем використовується для роботи із окремими клієнтами компанії у питаннях маркетингу, продажів, обслуговування і розробки нових продуктів. Використання таких систем спрощує процес взаємодії бізнесу із кожним окремим клієнтом, що у свою чергу дозволяє відкрити послідовні та активні відносини з ними.

Останній тип систем тісно пов’язаний із транзакціями обробки інформації в середині окремо взятої компанії. Такий тип систем пов’язує окремі відділи організацій, і контролює процес реалізації кінцевої продукції у відповідності із роботою кожного окремого відділу – продажі і маркетинг, продукція, фінанси, людські ресурси. У більших організаціях процеси обробки транзакцій інтегруються в окремі системи планування ресурсів підприємства (ERP), які

виступають ключовою основою чи не кожного підприємства. Такі системи виконують підтримку ланцюжка створення вартості продукту – повної послідовності дій або процесів, завдяки яким фірма додає вартість своїм продуктам. Принцип роботи таких систем схожий із розглянутими раніше SCM системами, але при цьому контроль за управлінням виробництва продукції виконується в рамках однієї організації, і всі відповідні її окремі підрозділи оновлюють дані щодо реалізації як тільки нова продукція була оформлена для нового клієнту компанії. З цього можна зробити висновок, що ERP системи можуть тісно співпрацювати із іншими різними інформаційними системами, щоб надавати окремій компанії деякий позитивний результат, як було зазначено серед вказаних ролей.

Рівень підтримки роботи із інформацією.

Наступний рівень розподілу інформаційних систем у організаціях пов’язаний із підтримкою роботи із інформацією. Цей рівень присвячений використання інформаційних систем із роботою, яка потребує роботу із деякою абстрактною інформацією або знаннями, а ніж роботою із прямою обробкою, розробкою і наданням готових даних за якимось відомими процесами. Така робота передбачає напівструктуроване прийняття рішень, адже дані при такому процесі можуть бути побудовані на деякому наборі настанов і правил, але при цьому потребують в необхідності видати остаточне рішення за користувачем системи. Через це, цей рівень також часто називають рівнем підтримки тактичного управління.

Серед категорій інформаційних систем, до цього рівня належать системи професійної підтримки, системи співпраці і системи управління знаннями.

Системи професійної підтримки використовуються в компаніях для виконання деяких завдань, які є специфічними до деякої визначеної професії. Наприклад, розробники програмного забезпечення в будь-якій організації інформаційних технологій використовують інтегровані середовища розробки (IDE) для написання програмного коду продукту, коли в той же час будь-який інший співробітник компанії не використовує функціонал таких програм.

Біохіміки використовують спеціалізоване програмне забезпечення для тривимірного моделювання для візуалізації молекулярної структури і тестування ймовірних ефектів нових препаратів перед виконанням клінічних тестів, тощо.

Системи співпраці використовуються для сприяння спілкуванню та роботі в команді між членами організації та між організаціями. Як вже було зазначено раніше, такий тип систем включає також декілька видів. Серед них можна назвати системи контролю спільним доступом, який використовується для управління процесом ведення відповідного проєкту, документу чи цілих комп'ютерних систем, при цьому співробітники можуть працювати за різними комп'ютерами у зовсім різних локаціях світу. Іншим типом систем співпраці можна назвати системи відеоконференцій або обміну миттєвими повідомленнями. За допомогою таких платформ можна виконувати конференції із віддаленими співробітниками, чи ділитися інформацією за допомогою засобів мережі Інтернет, використовуючи корпоративні месенджери.

Системи управління знаннями використовуються для збору, аналізу і запису корисної інформації і знань, отриманих в процесі роботи організації. Такі системи можуть допомогти у зборі необхідних знань про деякі процеси і правила, щоб у подальшому спростити необхідну кількість дій у роботі з деякими схожими задачами. Така інформація може включати в собі текст, зображення у поєднанні із документами, методами проєктування, найкращими практиками, тощо. Серед прикладів реалізації таких систем можна назвати використання корпоративної системи, яка працює за принципом вікі, яка дозволяє ділитися найкращими практиками знань із різними співробітниками організації. Крім того, слід зазначити, що організаційні знання часто є мовчазними, а не виразними, тому такі системи також повинні спрямовувати користувачів до членів організації, що мають спеціальний досвід.

Рівень підтримки управління.

Найвищий рівень інформаційних систем підтримки управління відповідає за допомогу над контролем організацією, і аналізу даних отриманих із результатів роботи систем на рівнях нижче, таких як системи транзакціями

процесів. Цей рівень використовується переважно менеджерами вищого рівня або керівниками окремих підрозділів цілих організацій, адже для керування такими системами їм необхідно виконувати неструктуровані рішення серед результатів внутрішніх інформаційних систем, і зовнішньої інформації, яка надається бізнес партнерами, постачальниками і клієнтами. Цей рівень ще також називають рівнем стратегічного управління.

Серед використовуваних систем цього рівня виділяють такі категорії: системи управлінської звітності, системи підтримки прийняття рішень, виконавчі інформаційні системи.

Системи управлінської звітності призвані для надання щоденних, детальних та об'ємних інформаційних звітів, які характерні для кожної сфери відповідальності менеджерів. Як правило, такі системи використовуються менеджерами або керівниками одного рівня управління, наприклад певним підрозділом, чи то відділом продажів, чи підтримки клієнтів. Менеджери користуються такими системами для аналізу інформації та продуктивності команди відділу за певний проміжок часу в минулому, або на поточний момент, і найчастіше не застосовуються для планування майбутньої продуктивності, чи використання в якості елементів для формування нових задач організації чи окремого підрозділу виробництва.

Системи підтримки прийняття рішень розроблені для спеціалізованого аналізу великого обсягу інформації для отримання деяких результатів для планування подальших планів розвитку організації у певному напрямку. Такі системи ще мають назву як програми бізнес-аналітики. Існують два окремих види таких типів інформаційних систем: на основі моделі і на основі наборів даних.

Перший вид систем прийняття рішень будується на принципі деякої запрограмованої моделі із відносно обмеженим набором даних. Наприклад, якщо в компанії необхідно закласти ціну на новий продукт, менеджер з продажу може скористатися системою прийняття рішень з маркетингу. У цю систему закладена деяка модель із різними змінними факторами – ціна продукту, вартість товару та

витрати на просування в різних засобах інформації – до прогнозованого обсягу продажів протягом перших п’яти років на ринку. Змінюючи фактори у самої моделі, менеджер зможе порівняти прогнозовані результати і вибрати вигідну ціну продажу.

У системах, побудованих на основі набору даних, закладений алгоритм аналізування великих об’ємів даних, які збираються протягом довгого періоду часу у сховищах даних. Така система намагається прогнозувати деяку статистичну модель на основі існуючих даних, і менеджери можуть користуватися отриманими результатами в якості цільових, за допомогою яких і зверталися до даної системи. Дуже часто такий тип інформаційних систем будується за допомогою технологій штучного інтелекту, так як навчена нейронна мережа зможе віднайти потрібну інформацію для прогнозування результатів серед великих об’ємів неструктурованих даних.

В якості виконавчих інформаційних систем виступають платформи для звітування великої кількості критичної інформації у дуже узагальненій і зручній формі, як правило, за допомогою графічної цифрової інформаційної панелі. Такі системи використовуються керівництвом компанії, такими як виконавчий директор, старший і виконавчий віце-президенти, рада директорів для моніторингу оцінки роботи компанії, оцінки ділового середовища та розробки стратегічних напрямків організації на майбутнє. Інформацією, яку оперує керівництво в рамках цих систем можуть бути як і звітності із усіх внутрішніх інформаційних систем меншого ранку, результати аналізу ринку із конкурентних організацій, в цілому значення економічних трендів на ринку. Використання таких систем дозволяє спростити процес побудови плану розвитку компанії у майбутньому, при цьому використовуючи лише узагальнену інформацію на поточний час [19, 40].

Як можна побачити, серед інформаційних систем в організації існує дуже багато окремих типів систем, які розподіляються по ролям які вони виконують, по типам даних які вони оброблюють для отримання результатів, які задовольнять користувача. Крім зазначених всіх типів і ролей інформаційних

платформ, важливо знати в якій організації і спеціалізації реалізовується конкретна система. Виходячи із заданих даних, необхідних для використання інформаційних систем в деякій спеціальності, визначаються її ключові ролі, і головні компоненти з яких вона має складатися.

Так як робота присвячена аналізу і розробці інформаційної системи у навчанні, перейдемо до розгляду видів інформаційних систем у освіті.

1.3. Види інформаційних систем в освіті

Інформаційні системи управління навчанням, до яких можна частково віднести системи управління школами або вищими навчальними закладами, так само реалізовують процес управління даними пов’язані із начальним процесом.

Суть інформаційних систем управління навчанням полягає у побудові єдиної платформи, на якій можна слідкувати за продуктивністю роботи студентів, викладачів, а також різних підрозділів, робити аналіз отриманого дослідження і на основі нього приймати деякі рішення [28]. Перевага використовувати інформаційні системи управління в навчанні полягає в тому, що в них можна реалізовувати дослідження і обробку даних серед різних секцій і окремих підсистем, пов’язаних із процесом навчання. Як приклад, можна назвати такі секції як: управління фінансами, зберігання персональної інформації про студентів або учнів, управління відвідуваністю, організація розкладу навчання, менеджмент навчального персоналу тощо. Кожний окремий навчальний заклад сам визначає які модулі слід реалізовувати в їх інформаційній системі, але чим більше додаткових функцій може бути створена в окремій платформі, тим потужнішою вона стає [31].

Як приклад, найпопулярніші модулі які реалізовують у сучасних інформаційних системах управління навчанням можна побачити на рис. 1.5:

Рис. 1.5. Можливості розвитку інформаційних систем управління навчальним процесом

1.4. Приклади реалізації інформаційних систем управління навчанням

Розглянемо деякі приклади реалізації інформаційних систем управління навчанням, в яких буде показано які можливості реалізовані в кожній окремій розглянутій системі.

Було перевірено ряд інформаційних систем, реалізованих у вищих навчальних закладах Росії. Серед них, можна назвати наступні зазначені інформаційні системи, та приклади їх реалізованих функцій [15]:

1. Інформаційна система «Управління якістю навчального процесу. Облік успішності і відвідуваності» Самарського державного технічного університету. Серед функціоналу було реалізовано можливість ведення успішності і відвідуваності студентів, формування списку спеціальностей і дисциплін всього університету, підтримка актуальності документів [13].

2. Інформаційна система «Освіта» Санкт-Петербурзького державного університету аерокосмічного приладобудування. Функціонал цієї системи доволі великий, який включає у собі також ведення списків спеціальностей і дисциплін, а також можливість формувати з них звіти, які готові для друку; журнали

відвідуваності і боржників за дисциплінами, формування списку відрахованих студентів і випускників; контроль за нарахуванням стипендії а також автоматизація роботи прийомної комісії, тощо.

3. Інформаційна система Волгоградського державного технічного університету. Серед функціоналу цієї системи можна назвати можливість збереження і правки нормативно-довідкової інформації про ВНЗ, збереження особових карток студентів, формування списків дисциплін і навчальних планів підготовки студентів, ведення звіту успішності, формування і нарахування стипендій.

4. Автоматизована інформаційна система Московського інституту сталі і сплавів. Ця система складається з окремих підсистем, які називаються «Управління контингентом студентів», «Звіт поточної успішності і відвідуваності», «Планування навчальним процесом», «Розрахунок стипендії». Відповідно до зазначених підсистем, у цій платформі реалізовані модулі картотеки студентів, із можливістю переглядати історію зміни даних про кожного і взаємодії з ним, формування навчальних планів відповідно до поточного навантаження із врахуванням існуючих пріоритетів, контроль за відвідуваністю і успішністю [16].

5. Автоматизована інформаційна система «Університет» Ставропольського державного університету. Серед можливостей зазначеної системи можна назвати контроль діяльності вищого навчального закладуі, перевірка і обробка даних студентів, робітників навчального закладу, науково-методичної діяльності його співробітників, розрахунок навчальних годин і можливість розподілу навчального навантаження і аудиторного фонду між викладачами з урахуванням їх побажань, аналіз і планування навчальних занять і екзаменаційної сесії, а також модуль збереження всеросійських і міжнародних класифікаторів і довідників, необхідних для проведення і публікації наукових досліджень.

6. Інформаційна система Смоленського гуманітарного університету (складається з двох елементів: інформаційно-телекомунікаційна мережа і інформаційний фонд). Серед переваг даної системи можна назвати можливість

збереження різної інформації, пов’язаної із навчальним процесом і необхідної для управління вищим навчальним закладом, а також автоматизація діяльності посадових осіб університету для вирішення управлінських задач.

7. Інформаційна система «Університ» Південно-Уральського Державного Університету. Ця система також включає такі підсистеми як «Учбова діяльність», «Студентські служби», «Управління персоналом». До можливостей зазначеної платформи можна вказати ведення обліку студентів і абітурієнтів при їх зарахуванні на навчальну програму; контроль переведення студентів на наступний курс, випуск і відрахування; планування навантаження викладачів і побудова графіку навчальних подій, системи сповіщення студентів, викладачів, абітурієнтів, батьків; збереження даних щодо розміщення студентів у гуртожитках; контроль і корегування бюджету навчального закладу щодо заробітної плати персоналу, підрахунок трудового стажу, зміна графіку робочого часу, тощо [14].

Серед інших прикладів розроблених платформ управління навчанням можна також назвати Інформаційно-аналітичну систему університету «Електронний університет» Казанського федерального університету. В зазначених джерелах вказано, що на поточний момент у системі також реалізовано декілька різних модулів для покращення процесу навчання. Серед них можна назвати модулі автоматизації управління навчальним процесом, які поділяються відповідно на абітурієнтів, студентів, аспірантів; також реалізовано персональні сторінки для студентів, викладачів і співробітників вищого навчального закладу для забезпечення робочого процесу і науково-дослідної діяльності; присутні складові автоматизації управління фінансово-господарських процесів забезпечення університету, включаючи програми нерухомості і гуртожитку для студентів тощо [11, 12].

Серед інших прикладів впровадження інформаційних систем управління навчанням можна назвати велику кількість платформ впроваджених у США. Одна з таких – інформаційна система Університету Колорадо, Боулдер, яка має назву «MyCUinfo». Така розроблена система надає наступний функціонал:

студенти мають можливість переглядати розклад занять, оцінки з дисциплін, отримувати останні оновлення щодо університету і гуртожитку, оплачувати навчання і проживання, передивлятися пропозиції щодо роботи, отримувати доступ до різних студентських сервісів, каталогу бібліотеки, а також перегляд користувацьких файлів. У свою чергу, викладачі і персонал університету мають можливість розміщувати оголошення в системі, змінювати розклад студентів, розміщувати навчальні матеріали, а також вести різну звітність, в тому числі фінансову [33].

Іншим прикладом реалізованої інформаційної системи є платформа «MyClarion» розроблена для Університету Кларіона, що у західній Пенсильванії. Задана платформа також реалізовує окремий функціонал для студентів, викладачів та іншого персоналу навчального закладу, і розподіляє платформу на три основні підсистеми: студентський центр, факультетський центр і рішення для кампусу. Студентський центр пропонує студентам можливість переглядати інформацію про навчання, оцінки за дисципліни, можливості отримати фінансову підтримку і сплачувати різні рахунки і платежі за навчання і проживання. Факультетський центр надає можливість викладачам і різному факультетському персоналу надавати доступ до редагування розкладів та списків занять, розставляти оцінки за дисципліни студентів, надання навчальної консультації тим, хто її потребує, тощо. Рішення для кампусу є розділом для працівників адміністративно-академічного відділу, і надає способи адміністрування усією платформою, із можливістю модифікувати всі студентські системні функції [32].

Висновки до розділу 1

Інформаційні системи є одним із ключових елементів сучасних структур будь-яких організацій у світі. На поточний момент такі платформи виконують велику кількість ролей для пришвидшення і покращення загальної продуктивності як і організації, так і окремих її компонентів, в тому числі її співробітників.

Якщо розглядати використання інформаційних систем в освіті, вони також грають не менш важливу роль для покращення в цілому процесу навчання, управління ключовими елементами навчальних закладів, а також ряду інших важливих компонентів. Серед основних використовуваних інформаційних систем в освіті в основному розглядають платформи реалізації дистанційного навчання, а також управління навчальним процесом, адже саме ці два важливих компоненти потребують покращення продуктивності обробки великої кількості даних, а також забезпечення учнів або студентів кращими засобами для навчання, або його контролю за ним.

Було виконано огляд сучасних популярних реалізацій інформаційних систем управління навчанням в ряді деяких вищих навчальних закладів. Кожна окрема система відрізняється як і реалізованим набором модулів, які дозволяють спростити деякий процес пов’язаний із навчанням, так і типом системи (комерційна або власна розробка) яка використовується в якості основи для роботи із заданим навчальним закладом. В рамках другого розділу буде розглянуто порівняння таких систем один з одним, і буде виокремлено ряд складових пов’язаних із навчально-методичним процесом.

Розглянуті відсутні можливості у відповідних системах буде вказувати на новизну розробки і їх можливість використання в рамках окремої інформаційної системи. Такі елементи будуть розглядатися як основа для наукової новизни даної роботи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ МОДЕЛІ НАВЧАЛЬНО-МЕТОДИЧНОГО КОМПЛЕКСУ В РАМКАХ ОКРЕМОЇ ПЛАТФОРМИ АВТОМАТИЗАЦІЇ НАВЧАЛЬНОГО ПРОЦЕСУ

2.1. Порівняння прикладів реалізації інформаційних системи навчання в різних закладах

Як було розглянуто у Розділі 1, де було наведено деяка кількість реалізації інформаційних систем у різних навчальних закладах світу. Кожна окрема система мала ряд своїх реалізованих функцій і особливостей, яким вона характеризується. Для визначення деяких важливих функцій, яких бракує у розглянутих прикладах необхідно виконати порівняльну характеристику розглянутих систем, і виділити їх реалізований функціонал.

Дану порівняльну характеристику побудовано у таблиці 2.1 з відповідними результатами [5, 6, 11, 12, 13, 14, 16]:

Таблиця 2.1. Порівняльна характеристика розглянутих прикладів інформаційних систем управління навчанням

Інформаційні системи	**Рішення комерційне, чи власна розробка**	**Можливості інформаційної системи управління**
1	**2**	**3**
Адміністративно-навчальна інформаційна система інституту комп’ютерних	Власна розробка	Поділ системи на два основних розділи: Адміністративний та Навчально-методичний. Перегляд кадрової інформації щодо кожного суб’єкту навчання; поділених на викладачів, студентів та аспірантів; формування особистих

Таблиця 2.1. Продовження

1	2	3
технологій Національного авіаційного університету України		карток співробітників, аспірантів та докторантів; фільтрація та пошук за обраними критеріями відповідного кадрового складу; формування та друк різноманітних звітів, щодо адміністративної інформації про інститут. Побудова навчальних та робочих планів за спеціальностями та напрямками підготовки; генерація окремим файлом отриманих навчальних і робочих планів. Збереження навчального та навчально-методичного матеріалу для студентів, щодо виконання будь-яких атестаційних робіт та контролю знань та умінь; перегляд наукових та навчально-методичних видань співробітників інституту; пошук та фільтрація серед збереженого навчального та наукового матеріалу.
Інформаційна система управління навчальним процесом Донецького національного технічного університету	Власна розробка	Функціонал для студентів: можливість перегляду поточної успішності і відвідуваності занять; вибір необов’язкових дисциплін; перегляд розкладу поточного семестру; доступ до університетських новин, подій, пошук пропозицій роботи; облік заборгованостей по бібліотеці; доступ до навчальних матеріалів. Функціонал для викладачів: контроль і внесення поточних даних про успішність і відвідуваність

Таблиця 2.1. Продовження

1	2	3
		студентів; додавання навчальних матеріалів; розміщення оголошень і новин. Функціонал для абітурієнтів: перегляд поточного рейтингу; доступ до навчальних матеріалів з підготовки до університетських дисциплін; міграція у роль студента при успішному вступі.
Інформаційна система «Управління якістю навчального процесу. Облік успішності і відвідуваності» Самарського державного технічного університету	Власна розробка	Облік і аналіз успішності і відвідуваності студентів. Формування і ведення списку спеціальностей і дисциплін усього університету; оперативне отримання актуальної інформації в автоматичному режимі. Підтримка актуальності документів.
Інформаційна система «Освіта» Санкт-Петербурзького державного університету аерокосмічного	Власна розробка	Збереження, додавання, коригування списків, документів; введення журналів успішності; перегляд даних списків у вигляді карточки; пошук по будь-якому параметру; фільтр (вибірка даних) за будь-яким параметром; впорядкування. Друк стандартних документів; введення,

Таблиця 2.1. Продовження

1	2	3
приладобудування		збереження і корегування інформації про студентів, які проживають у гуртожитку; друк довільних документів із зазначенням будь-яких параметрів; формування і друк додатків до диплому за затвердженим зразком. Журнал успішності; відомість боржників; облік нарахування стипендії; ведення списку студентів у академвідпустці, відрахованих студентів і випускників; переведення на наступний курс; формування звітів для пенсійного фонду; автоматизація роботи прийомної комісії; формування і друк екзаменаційних відомостей. Процедура резервного копіювання даних.
Інформаційна система Волгоградського державного технічного університету	Власна розробка	Збереження і корегування нормативно-довідникової інформації про ВНЗ (факультети, кафедри, напрямки освіти); збереження і корегування особистих карточок студентів; списків дисциплін по контрольних тижнях, успішності студентів під час сесії, учбових планів підготовки студентів; Моніторинг успішності студентів; ведення обліку поточної успішності в учбовій групі, відомостей по успішності студентів у сесію, облік нарахування та призначення стипендії).
Автоматизована інформаційна	Комерційне рішення	Відстеження контингенту учнів у відповідних структурних підрозділах і службах ВНЗ; ведення

Таблиця 2.1. Продовження

1	2	3
система Московського інституту сталі і сплавів		картотеки учнів, обліку історії змін даних про кожного учня і історії взаємодії з ним. Формування стандартних форм звітності, збереження результатів рейтингового тестування абітурієнтів із можливістю формування конкурсного ряду; Оформлення та коригування учбових планів за напрямками з врахуванням існуючих в інституті пріоритетів; розрахунок учбового навантаження по кафедрам; контроль поточної успішності студентів і відвідуваності.
Автоматизована інформаційна система «Університет» Ставропольського державного університету	Власна розробка	Загальний моніторинг діяльності ВНЗ; управління якістю навчального процесу; збереження і обробка інформації про студентів і робітниках ВНЗ, науково-методичної діяльності співробітників. Розрахунок учбових годин, розподіл учбового навантаження та аудиторного фонду з урахуванням побажань викладачів, розташування корпусів університету і особливостей процесу навчання; оптимізація штатного складу; збереження і аналіз планів-графіків, розкладу учбових занять і екзаменаційних сесій; формування розкладу занять. Збереження бази даних, яка містить всеросійські

Таблиця 2.1. Продовження

1	2	3
		і університетські класифікатори і довідники, необхідні для проведення наукових досліджень і публікації наукових досягнень.
Інформаційна система Смоленського гуманітарного університету (складається з двох елементів: інформаційно-телекомунікаційна мережа і інформаційний фонд)	Власна розробка	Збереження різної інформації, яка використовується в учбовому процесі, науково-дослідної діяльності і необхідної для управління ВНЗ. Автоматизація діяльності посадових осіб університету при вирішуванні управлінських задач.
Інформаційна система «Університ» Південно-Уральського Державного Університету	Власна розробка	Різні види планування і розробок; збереження і корегування учбових планів; збереження списку абітурієнтів і зарахованих студентів на навчальну програму, облік допуску до контрольних заходів; облік успішності; облік студентів, переведених і непереведених на наступний курс; збереження і зміна даних про студента, в тому числі про випуск і відрахування студента; планування навантаження, планування і побудова графіку учбових заходів. Інформування абітурієнтів, батьків, спонсорів за

Таблиця 2.1. Продовження

1	2	3
		допомогою веб технологій і смс технологій; облік оплати за навчання студента; збереження даних про розміщення студентів у гуртожитку, можливість вирішення питань роботи гуртожитку. Ведення військового обліку студентів і робітників ВНЗ; ведення штатного розкладу; збереження і корегування видаткової частину бюджету з оплати праці; управління даними окладів; відображення змін і ведення історії зміни даних по персоналу; рахунок трудового стажу; відслідковування даних по відпусткам; реєстрація відпускних днів, збереження даних про графік робочого часу; збереження інформації про правила, процедури нагородження і заохочення робітників ВНЗ.
Інформаційно-аналітична систему «Електронний університет» Казанського федерального університету	Власна розробка	Автоматизація управління навчальним процесом для абітурієнтів, студентів, аспірантів ВНЗ; інформаційне забезпечення робочого процесу і науково-дослідної діяльності користувачів, персональні кабінети співробітника ВНЗ, студента і викладача. Автоматизація управління фінансово-господарських процесів забезпечення діяльності ВНЗ, окремі підпрограми роботи із гуртожитками системи, вибір і оренда

Таблиця 2.1. Продовження

1	2	3
		нерухомості для студентів і співробітників, програма розвитку університету. Забезпечення актуальної інформації про університет, надання веб-сервісів для співробітників і студентів університету, здійснення інтеграції із модулями окремої інформаційно-аналітичної системи «Портал КФУ». Підтримка процесу забезпечення структурних підрозділів безперебійною роботою комп'ютерної техніки та програмного забезпечення Департаменту інформатизації і мережі університету.
Інформаційна система «myCUinfo», Університету Колорадо, Боулдер.	Комерційне рішення	Функціонал студентів: перегляд розкладу занять, ведення успішності за окремими дисциплінами, щоденні новини про заклади, які проводяться в університеті та гуртожитку, можливість оплачувати навчання в університеті та проживання в гуртожитку, отримання інформації про пошук підробіток, доступ до великої кількості студентських сервісів доступних на території університут, включаючи можливість обрати літературу в каталозі бібліотеки, ведення, збереження і корегування користувацьких файлів студентів, і можливість ними ділитися.

Таблиця 2.1. Продовження

1	2	3
		Функціонал викладачів і персоналу університету: можливість розміщення оголошень про університет, окремі дисципліни і групи студентів, збереження і корегування розкладу занять студентів, збереження і розміщення навчальних матеріалів дисциплін, ведення різної звітності, в тому числі успішність кожного окремого студента, групи, а також фінансова інформація.
Інформаційна платформа «myClarion» Університету Кларіон, західна Пенсільванія	Комерційне рішення	Функціонал студентського центру: перегляд інформації про навчання, що включає в собі отримання звітності про поточну успішність і оцінки за дисципліни, перегляд розкладу занять, можливість отримати фінансову підтримку у ряді відповідних питань, сплата за навчання в університеті і проживання у гуртожитку. Функціонал факультетського центру: перегляд, редагування і корегування розкладів і списків занять у окремих груп студентів, перегляд і редагування оцінок за дисциплін студентів, можливість надання навчальної консультації студентам із низькою успішністю. Функціонал центру рішень для кампусу: адміністрування платформи в цілому, додання окремих нових модулів в процес навчання інших двох підсистем, управління адміністративно-.

Таблиця 2.1. Продовження

1	2	3
		академічним відділом університету.

Як можна побачити із результатів таблиці, багато систем мають свій набір реалізованих функцій, які досить відрізняються один від одного в залежності від потреб університету. Також, можна побачити що комерційні рішення в середньому пропонують більше функцій для реалізації інформаційних систем у навчанні, але як правило реалізація таких окремих систем в окремих університетах потребує більше фінансових затрат за більшу кількість додаткових модулів пов’язаних із навчанням. Серед таких додаткових модулів можна вважати фінансові операції пов’язані із системою, можливість роботи із трудовою складовою штату університету, обробка даних пов’язаних із проживанням у гуртожитку, тощо.

Так як поточна робота зосереджена із реалізацією навчально-методичного комплексу в рамках окремої інформаційної системи автоматизації навчального процесу, в подальшому продовжимо дослідження важливих функцій навчання, реалізація яких в окремій електронній системі значно спростить ведення навчального процесу в деякому зазначеному вищому навчальному закладі.

2.2. Сутність моделі навчально-методичного комплексу в навчальних закладах

Перед розглядом суті терміну навчально-методичного комплексу, розглянемо детальніше розуміння його основного компоненту – процесу навчання в цілому, як основного елемента освітнього процесу в навчальних закладах.

В якості процесу навчання найчастіше розглядають процес пізнавальної діяльності, під час якої учень отримує необхідні знання від викладача в де-якій предметній області. Під час процесу навчання учень засвоює отриману

інформацію від вчителя, робить висновки щодо отриманих навичок, і в подальшому матиме можливість їх реалізовувати у професійній діяльності в заданому предметі [3].

Якщо окремо розглядати процес навчання у вищих навчальних закладах (ВНЗ), він має свої особливості реалізації, які відрізняються як на рівні університету, так і на рівні окремих факультетів і кафедр. Правила які узагальнено регулюють процес навчання будуються в рамках окремих нормативних документів, які спеціально диктують основні положення навчання на рівні університету.

Як приклад, візьмемо положення про навчальний процес в НТУУ «КПІ ім. Ігоря Сікорського» [8]. В рамках цього документу визначено ряд окремих термінів, на основі якого будується навчання на всіх факультетах і кафедрах університету, за допомогою якого можна отримати вищу освіту. Розглядаючи багато інших джерел, в яких описано процес побудови навчального процесу в різних навчальних закладах, або організація того ж навчально-методичного комплексу, можна виділити ряд схожих термінів і процесів, які не змінюються від одного ВНЗ до іншого. Саме завдяки схожій структурі організації навчального процесу виходячи з однакових термінів реалізації навчання можна організувати узагальнену модель навчально-методичного комплексу на їх основі.

Розглянемо детальніше основні елементи, на основі якого формується узагальнена модель такого комплексу, виходячи із прикладу документу – положення про навчальний процес у НТУУ «КПІ ім. Ігоря Сікорського».

2.2.1. Опис моделі планування освітнього процесу

В якості основи процесу навчання в університеті виступають *освітні програми*. Вид такої програми залежить від необхідних вимог до рівня освіти майбутнього здобувача, переліку освітніх компонентів та послідовності їх викладання, кількості кредитів ЄКТС, які необхідні для виконання цієї програми,

а також очікувані результати навчання. Так, в цілому розглядають наступні рівні вищої освіти, на основі яких формуються освітні програми:

- перший (бакалаврський) рівень;
- другий (магістерський) рівень;
- третій (освітньо-науковий) рівень;
- науковий рівень.

Крім освітньої програми окремі структури університету регламентують *графік навчального процесу*, під час якого забезпечується навчання здобувачів освіти. Як правило, графік визначає часи проведення навчальних занять, екзаменаційних сесій, практик атестацій і канікул, і в цілому залежить від окремих спеціальностей, а також форм навчання студентів.

На основі зазначених двох елементів формується нормативний документ, що має назву *навчальний план*. Цей план передбачає перелік і обсяг навчальних дисциплін (предметів викладання), їх послідовність, а також розподіл навантаження навчальних годин за заняттями і формою атестації. Особливість зазначеного документу полягає в тому, що він складається лише раз на весь термін навчання певної групи студентів, і прямо пов’язаний з їх освітнім процесом.

З метою уточнення змісту навчання і закріплення навчальних дисциплін, до навчального плану складається окремий документ – *робочий навчальний план*. Він виконується окремо кафедрами, відповідними за освітню програму кожен раз на наступний навчальний рік.

Крім зазначених вище двох видів навчальних планів, існує також поняття *індивідуального навчального плану*. Він передбачає позицію здобувача освіти із вибору необхідної кількості проходження навчальних дисциплін, їх видів, форм і темпу проходження. Здебільшого, цей план формується також раз на кожен рік навчання.

Слід також врахувати, що для регламенту організації освітнього процесу, на основі побудованих робочих навчальних планів формується *розклад навчальних занять*. Розклад з-де більшого залежить від кількості здобувачів освіти, які

здають поточну робочу навчальну програму а також від графіку завантаження аудиторій і лабораторних кімнат.

Організація освітнього процесу виконується за наступними формами:

- навчальні (аудиторні) заняття;
- самостійна робота;
- практична підготовка;
- контрольні заходи.

Навчальне заняття передбачає процес проведення навчального процесу для здобувача разом із викладачем дисципліни у аудиторному, або дистанційному форматі для отримання основної інформації про предмет. Розглядають наступні види навчальних занять:

- лекція – основна форма проведення навчальних занять, призначених для засвоєння теоретичного матеріалу. Як правило, проводиться в усній формі у вигляді публічного читання;
- лабораторне заняття – вид проведення навчальних занять, при якій учень має змогу практично підтвердити здобутий теоретичний матеріал у вигляді деякого експерименту в лабораторних умовах;
- практичне, семінарське заняття – форма навчального заняття, при якій виконується дискусія між викладачем і учнем щодо деякої теми за навчальною дисципліною, з ціллю практичного засвоєння теоретичного матеріалу;
- індивідуальне навчальне заняття, консультація – вид проведення навчальних занять, який проводиться з метою підвищення рівня підготовки в зазначеній навчальній дисципліні.

Самостійна робота виступає як засіб засвоєння навчального матеріалу в окремий час від навчальних занять. Він передбачає опрацювання навчального матеріалу, виконання домашніх і індивідуальних завдань, підготовку до лекцій, або практичних занять, тощо. В якості індивідуальних завдань можуть виступати *курсові роботи або проєкти*, які контролюють деякий творчий процес проведення дослідження на окрему тему в рамках навчальної дисципліни.

Практична підготовка або *практика* передбачає ознайомлення із засобами і методами роботи із знаряддями праці в галузі тої професії, яка прямо пов'язана із освітньою програмою здобувача освіти. Як правило, для здобуття таких навичок, студентів направляють в окремі підприємства, установи, організації, або окремі структурні підрозділи університету, за наявності необхідного устаткування.

Контрольні заходи є видом проходження контролю поточних знань та навичок, набутих в процесі навчання з метою підтвердження здобутих знань і умінь на момент проходження таких заходів. Існують декілька видів контрольних заходів для перевірки знань – *вхідний, поточний, календарний, підсумковий* (екзаменаційна сесія та атестація) і *ректорський*. В якості оцінювання цих заходів університет надає свою рейтингову шкалу, яка підтверджує рівень здобутих знань в процесі вивчення дисципліни. На прикладі системи РСО, максимально можливий бал є 100, а мінімально допустима оцінка з проходження дисципліни – 60.

Крім зазначених раніше важливих елементів освітнього процесу в рамках вищого навчального закладу, важливими складовими організації з підготовки навчання виступають *організаційне та навчально-методичне забезпечення*. До їх складу входять *документи деканату і кафедр, документи зі спеціальностей, освітніх програм та навчально-методичне забезпечення*. Дані документи призвані надати процесу навчання конкретного змісту в рамках деканату чи кафедри, або відносно окремої навчальної дисципліни, яка врахована у конкретній освітній програмі [8].

Терміни, показані в цьому підрозділі, формують основу введення навчального процесу в рамках вищих навчальних закладів, зокрема НТУУ «КПІ ім. Ігоря Сікорського». Введення великої кількості різної документації на різних навчальних структурах, таких як кафедри, деканат, ректорат та інші, доволі ускладнює процес забезпечення коректної організації навчання. Крім того, не виключно, що у процесі підготовки відповідних документів і складових, необхідних для правильної роботи навчання, будуть виникати помилки,

неточності, неузгодженні між різними підрозділами університету, або помилки, пов’язані із контролем якості навчальних матеріалів. Ці питання призвані вирішуватися із модернізацією освітнього процесу, із введенням інформаційних системи управління освітою в цілому.

Як частину модернізації контролю якості вивчення навчальних дисциплін, і можливості управління процесом навчання дистанційно через комп’ютерні засоби, було створено поняття електронних навчально-методичних комплексів, або скорочено ЕНМК.

2.2.2. Інтеграція моделі планування навчального процесу в окремій інформаційній системі

Розглянемо тлумачення терміну електронно навчально-методичного комплексу докладніше. Сам термін має багато різних визначень від різних авторів, але розглянемо наступні два популярні терміни:

Електронний навчально-методичний комплекс (ЕНМК) – дидактична система, у якій з метою створення умов для педагогічної активності інформаційної взаємодії між викладачами і студентами інтегруються прикладні програмні продукти, бази даних, а також інші дидактичні засоби і методичні матеріали, які забезпечують і підтримують навчальний процес [7].

Електронний навчально-методичний комплекс (ЕНМК) – самостійне електронне видання, що складається із сукупності взаємодоповнюючих і взаємозв'язаних гіперпосиланнями дидактичних (програмних, теоретичних, практичних, методичних, наочних, довідкових, контрольно- вимірювальних і бібліографічних) засобів навчання за дисципліною навчального плану спеціальності, представлених на електронних носіях інформації, що сприяють ефективному засвоєнню навчального матеріалу дисципліни, необхідних для організації і здійснення самостійної роботи студентів, у тому числі – у системі дистанційної освіти [1].

Як можна побачити із визначень, сам по собі ЕНМК можна представляти собі як окрему інформаційну систему управління навчанням, яка у поєднанні з

комп'ютерними технологіями надає можливість набагато спростити процес введення освітнього процесу у навчальних вузах, ввести стандартизовану оцінку якості навчання. Крім того, зазначений комплекс надає можливість легко модифікувати процес вивчення деякої дисципліни, контролювати оцінку отриманих знань і умінь, а також дуже часто формувати необхідні звіти-результати вивченого навчального матеріалу.

Для правильної реалізації ЕНМК в рамках окремих навчальних закладів, вони мусять відповідати деяким зазначеним освітньо-кваліфікаційним стандартам. Крім того, для правильної подачі навчального матеріалу в рамках окремої дисципліни, кожна окрема освітня програма за спеціальністю повинна бути інтегрована у ЕНМК із відповідними затвердженими навчальними планами, щоб досягнути єдиної структури даних як і у інформаційному середовищі, так і зазначених у відповідних документах.

ЕНМК можна поділити на декілька компонентів, з яких вона складається. Серед них можна назвати такі блоки як програмно-інформаційний, навчально-методичний, контрольний, навчально-дослідницький, допоміжний. Розглянемо кожний із цих блоків детальніше.

Програмно-інформаційний блок ЕНМК.

Цей блок присвячений загальній структурі ЕНМК. В ньому зазвичай описується деякі відомості про автора, кафедру, дисципліну, спеціальності університету поточного плану роботи, зміст комплексу, тощо. В цілому, цей блок описується окремими навчальними планами, для яких формується навчально-методичний комплекс для вивчення окремих дисциплін.

Навчально-методичний блок ЕНМК.

Цей блок описує основу для подачі матеріалів за навчальною програмою. Він містить змістовні модулі, які наповнюються теоретичними (лекційними), навчально-практичними (завдання для опрацювання), методичними матеріалами (методична література, вказівки для виконання лабораторних, самостійних робіт, план і перелік питань для семінарів та інше), запитання для самоконтролю та перелік літератури.

Контрольний блок.

Контрольний блок комплексу складається з матеріалів, які необхідні для проведення контролю успішності отриманих знань та умінь студентів. Він містить перелік запитань до іспиту, підсумкові тестові завдання.

Навчально-дослідницький блок.

Цей блок містить список матеріалів і даних, необхідних для проведення окремих творчих або наукових робіт студентами. Ці матеріали включають набір тем для виконання кваліфікаційних робіт (курсові роботи і проєкти), рефератів, написання доповідей або статей з дослідження, тощо.

Допоміжний блок.

Цей блок містить додаткові матеріали, які можуть бути необхідні в процесі навчання. До таких матеріалів можна віднести різні мультимедійні та відео-, аудіо-файли, дослідні програми для виконання лабораторних або проєктних робіт, заготовки і моделі для виконання самостійної роботи, тощо [1].

Реалізація ЕНМК на поточний час потребує виконання ряду необхідних вимог, яка дозволить модернізувати навчальний процес у кращу сторону, і спростити доступ до вивчення матеріалу. Серед них, можна назвати наступні важливі вимоги реалізації електронних навчально-методичних комплексів:

- Доступність – інформаційна система має бути легко доступною для її користувачів через ряд різних каналів зв’язку;
- Адаптивність – розроблена система мусить мати однаковий інтерфейс і набір компонентів, який дозволить в ній інтегрувати можливість введення успішності за різними навчальними дисциплінами, мати можливість систематизувати навчальний матеріал між предметами і різними спеціальностями;
- Системність – система має відповідати освітньому стандарту введення навчального процесу, мати структуру відповідності освітній програмі, забезпечувати логічну послідовність засвоєння студентами матеріалів для вибраної дисципліни, підтримувати різні форми навчання, види занять, форми перевірки контролю і самоконтролю отриманих навиків;

- Мобільність – система може мати можливість налаштування доступу до даних які в ній містяться в залежності від засобів і способу доступу до комплексу.

Процес навчання у ЕНМК можна організувати за характером взаємодії між учнем і викладачем. Так, в режимі «інтерактивної» роботи студент має змогу консультуватися із викладачами у дистанційному форматі через вибраний канал зв'язку. В режимі «автоматизованої» роботи, студент проходить навчальний процес у автоматичному виді, в якому система моделює процес взаємодії із викладачем, використовуючи надані теоретичні і практичні матеріали.

Отже, маючи розглянуту структуру складових ЕНМК, а також її основних режимів роботи і задач, можемо побудувати узагальнену структурно-функціональну схему інтеграції навчального процесу у модель використання інформаційної системи. Отриману схему зображено на рис. 2.1.

2.3. Постанова проблеми розробки навчально-методичного комплексу в рамках інформаційної платформи автоматизації навчального процесу

Розробка розглянутого типу інформаційної системи ЕНМК, та її інтеграція в повноцінну комплексну систему автоматизації навчальним процесом в першу чергу полягає у узгодженні кінцевої моделі отриманого програмного продукту.

Задана система автоматизації навчальним процесом передбачає введення декількох важливих компонентів, які в цілому організовують процес роботи і вирішення бізнес процесів, пов'язаних із діяльністю університету. Архітектурна модель включає в собі декілька важливих компонентів для реалізації цих процесів, частиною якої має виступити і реалізація електронного навчально-методичного комплексу, для вирішення питань із організації навчального процесу здобувачів освіти, які будуть користуватися даною платформою.

Рис. 2.1. Структурно-функціональна схема інтеграції ЕНМК у процес навчання [9]

В основі самої платформи виступатиме централізована база даних, яка буде виконувати задачу збереження інформації про кожний окремий компонент та виконувати необхідні транзакції із обробки старих чи нових даних із окремими її сутностями. При вирішуванні задачі побудови такої бази даних необхідно врахувати важливе питання масштабованості усієї системи в цілому, так як із ростом кількості даних у базі, збільшенням потоку користувачів у платформу, зростає і навантаження на їх програмну і апаратну складову. Необхідно правильно і комплексно підійти до питання проектування моделі бази даних як основи всього проєкту для вирішення задачі масштабованості і розподілу апаратних і програмних її можливостей.

Перш ніж побудуємо модель представленої архітектури платформи на основі поставлених задач і окремих її компонентів, розглянемо в цілому існуючі моделі представлення архітектури комплексних систем.

2.3.1. Опис існуючих моделей архітектури побудови складних інформаційних систем для вирішення задачі масштабування

Перед тим, як перейдемо до існуючих моделей реалізації архітектури комплексних систем, розглянемо поняття масштабованості детальніше, і як її можна реалізувати у різних програмних продуктах.

Масштабування є неодмінним атрибутом деякої системи, мережі, чи процесу. Концепт цього питання полягає в тому, як де-яка система може пристосуватися до зростаючої кількості об’єму навантаження, або/чи бути сприйнятливою до розширення. При розробці нових програмних систем дуже часто виникає питання реалізації такого її дизайну, який дозволить їй бути легко масштабованою.

Якщо розглядати деяку систему, яка не є масштабованою, то найчастіше розуміють її неможливість справлятися із зростаючим навантаженням трафіку, або кількості обробленої інформації, яка на поточний момент зберігається в ній. Ціна цього значення може бути виміряна завдяки багатьом факторам, які можуть включають у собі обмежений час на відповідь, накладні витрати на обробку,

пам’ять, простір і навіть гроші. Великі показники витрат на обмеженість масштабованості доволі негативно впливає на якість обслуговування системи, і може призвести до затримок чи позбавлення можливостей доходу від проєкту [18].

Існують два підходи реалізації масштабованості в комп’ютерних системах, які відповідно мають назви горизонтальне і вертикальне масштабування.

Горизонтальне масштабування у своїй концепції полягає збільшення інфраструктури системи за для збільшення її паралельної роботи. При такому підході реалізовані апаратні можливості інформаційної системи розглядають як окремий обчислювальний блок. За рахунок правильного планування додатку системи для роботи її у паралельному режимі, збільшення кількості окремих обчислювальних блоків також збільшує і продуктивність системи в цілому. При такому підході ріст апаратної складової для інфраструктури залежить лінійно від кількості доданих окремих обчислювальних блоків, в той же час продуктивність додатку системи не матиме лінійної залежності від початкових значень, і буде відповідати програмній архітектури додатку.

Вертикальне масштабування полягає у рості загальної продуктивності системи за рахунок заміни поточного апаратного забезпечення більш потужним. При такій концепції із рівномірним ростом продуктивності додатку системи, ми не будемо мати лінійну залежність від потужності апаратної складової системи. Крім того, такий вид масштабування може призвести до проблеми, коли вже не існує більш потужних апаратних компонентів ніж існуючі, і подальший ріст буде вже неможливий [17].

Розглянуті концепції масштабування по різному проявляють свою ефективність в різних моделях програмних додатків. На поточний час існують три моделі представлення архітектури складних інформаційних систем, які відрізняються як і підходом до планування розподілу ресурсів, управління бізнес логікою проєкту, а також створення окремих програмних компонентів, в тому числі інтерфейс користувача:

- Монолітний підхід;

- N-шарова модель;
- Архітектура мікросервісів.

Для комплексного підходу вирішення підбору коректної архітектури у відповідності до питання масштабування, переглянемо кожну з них окремо.

Монолітний підхід.

Підхід до розробки монолітних програмних рішень є одною із перших концепцій розробки в цілому. Суть такого виду архітектури полягає в тому, що всі структурні компоненти платформи розміщуються в одному сервісі-місці, включаючи інтерфейс користувача, бізнес логіка продукту і базу даних. Схематично це можна представити як на рис. 2.2:

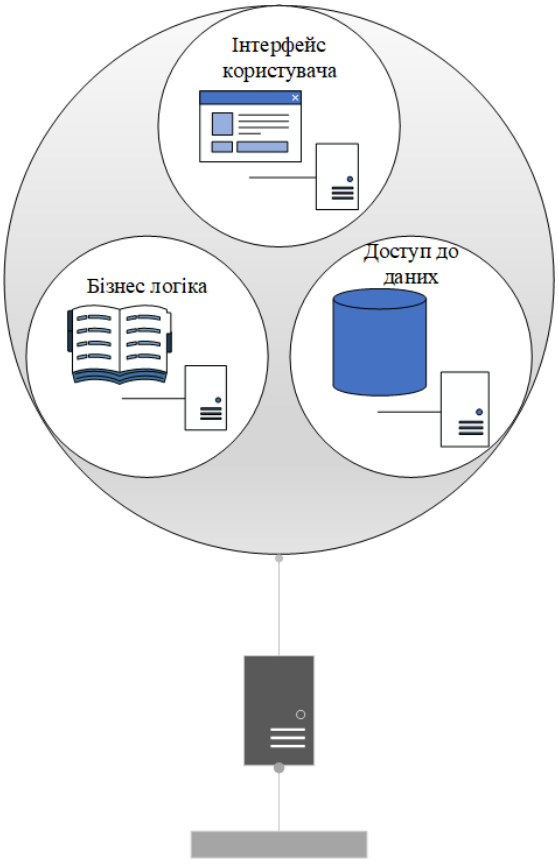


Рис. 2.2. Монолітна архітектура комплексної інформаційної системи

Серед переваг використання такої архітектури для розробки власної інформаційної системи можна назвати:

- Легка реалізація і тестування моделі на практиці;
- Простота розгортання моделі;
- Можливість просто масштабувати горизонтально, використовуючи в архітектурі Load Balancer.

Крім зазначених переваг, цей вид системи також має ряд і недоліків:

- Необхідність розгортання кожний раз, як виходить нове оновлення на додаток;
- Технічна підтримка при великому обсязі програмного додатку в цій моделі;
- Надійність системи, при якому при будь-якій критичній помилці в окремому моделі може зламати весь процес;
- Складність масштабування якщо різні модулі мають різні запити на використання ресурсів

Серед розглянутих переваг і недоліків, монолітний підхід виступає корисним коли існує потреба у побудові невеликих інформаційних систем, які не будуть отримувати дуже часто оновлення, і за потреби можуть масштабуватися, якщо поточних обчислювальних результатів недостатньо. Якщо існує необхідність у розробці платформи з різним функціоналом, потребуючим дуже часте оновлення, а також більш гранульований контроль за кожним окремим модулем в середині системи, слід розглянути інші шаблони [38].

N-шаровий підхід.

На відміну від моноліту, N-шарова модель в якості своєї основи полягає у розподілі програмних компонентів між деякими логічними шарами. Типова конфігурація системи в такому підході включає 3 основних шари -Рівень представлення, на якому реалізовується інтерфейс користувача; Рівень логіки. Який включає основну бізнес логіку програми; Рівень даних, який надає дані для обробки, на якому реалізована робота бази даних. Приклад такої моделі показано на рис. 2.3.

Серед переваг такої архітектури можна вказати наступні її елементи:

- Простота у розробці, так як кожний окремий шар може бути розроблений окремо один від одного різними командами розробників;
- Тестування додатку при такій архітектурі через ізольованість окремих шарів є доволі простою задачею;

- Шари можуть додаватися і відніматися за необхідності гранульованості структури додатку.

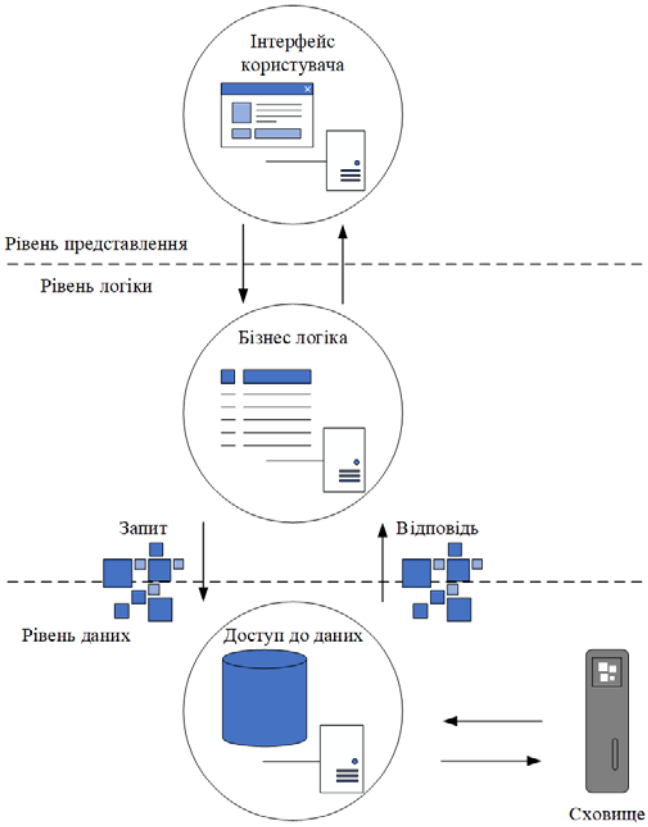


Рис. 2.3. N-шарова архітектура комплексної інформаційної системи

Серед недоліків такої моделі представлення додатку виділяють такі елементи:

- Складність розгортання. Хоча розробка шарів ізольована одна від одної, кожне велике оновлення окремого компоненту все ще потребує розгортання всього додатку;
- Продуктивність додатку при використанні у великих додатках. Ця проблема пов’язана із використанням моделі шарів, так як для обробки бізнес запитів необхідно пройти через усі шари архітектури;
- Складність масштабування. Через неоднорідність шарів між собою, контролювати масштабування системи в цілому складно, і потребує більших затрат у порівняні з іншими моделями.

Розглянута архітектура дозволяє надати більше контролю розробникам програмного продукту і розподілити їх обов’язки між окремими шарами, але при

цьому має все ще ряд важливих недоліків. Наступна модель дозволяє усунути їх, при цьому мати ряд інших плюсів [36].

Мікросервісна архітектура.

Мікросервісний підхід до розробки комплексних систем з'явився як вид для вирішення недоліків програм, побудованих на основі монолітної архітектури. Для цього, в основі цього підходу використовується розподіл основного функціоналу інформаційної системи за деякими модулями-сервісами, які можуть виконувати свою задачу незалежно один від одного. Всі модулі підключаються до єдиного інтерфейсу користувача, таким чином формуючи остаточний результат обробки даних.

Виходячи з цього, мікросервіси використовують модель розподілених додатків у поєднанні із структурою компонентів додатку при монолітній архітектурі, і на основі цього формують власний тип представлення принципу обробки даних в деякій комплексній системі. Схематично, мікросервісна архітектура показана на рис. 2.4:

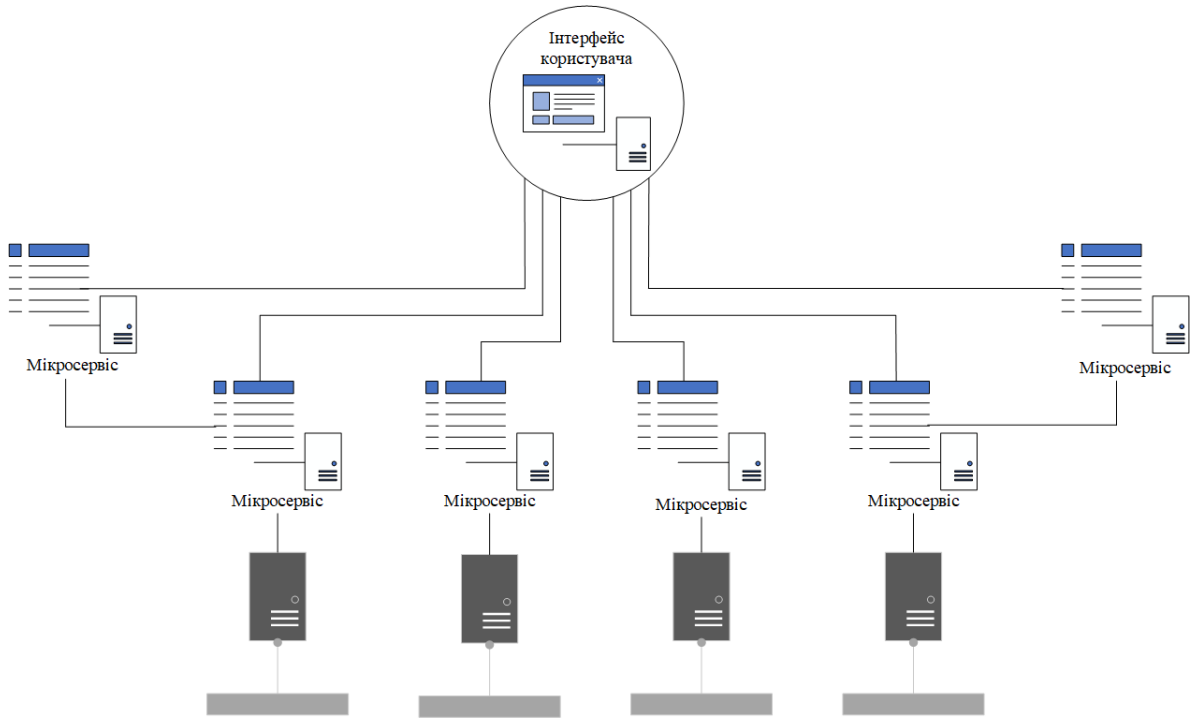


Рис. 2.4. Мікросервісна архітектура комплексної інформаційної системи

До переваг зазначеної архітектури вказують наступні її властивості:

- Розгортання системи. Так як кожний окремий функціонал в такій моделі реалізовується на окремому модулі, стає просто управляти оновленнями компонентів, і залишати систему у робочому стані;
- Розробка додатку. Так як кожний функціонал системи розбитий на окремі модулі, задача створення нових компонентів є простішою через їх незалежність один від одного. Це також стосується використання різних сховищ даних для їх обробки окремими компонентами;
- Тестування додатку, і окремих його компонентів. Через незалежність функціоналу один від одного, можна виконувати відповідні тести для окремих компонентів, і при цьому залишати систему робочою;
- Масштабування окремих компонентів. Через гранулювання окремих компонентів, можна за необхідності легко збільшувати апаратні можливості відповідного функціоналу за потреби збільшення продуктивності, незалежно від інших компонентів.

Але, крім зазначених переваг, у цій архітектурі також існують і недоліки:

- Продуктивність системи. Так як кожний окремий модуль працює незалежно і паралельно один від одного, продуктивність не може бути надто високою через розподілену природу цієї моделі;
- Проєктування системи зі збільшенням функціоналу. Так як кожний окремий функціонал потребує реалізацію нового окремого модуля, складність проєктування зростає із ростом кількості окремих сервісів;
- Високе навантаження мережі. Як наслідок використання розподіленої архітектури, через збільшення кількості взаємодії між окремими мікросервісами системи, збільшується і затримки в мережі через передачу даних [36, 38].

Як можна було побачити, серед розглянутих трьох видів архітектури комплексних систем, кожна з них має свої переваги і недоліки. При цьому, можна вказати що кожна з них може використовуватися для розробки власного додатку, в залежності від окремих потреб.

Так як основною метою для розробки заданої інформаційної платформи постає питання масштабування для підбору оптимального навантаження на окремі компоненти, а також розбиття функціоналу за окремими системами, то найкращий підхід до розробки такого продукту відповідає мікросервісній архітектурі. В подальшому, зосередимося на представленні архітектури платформи, і побудові структури навчально-методичного комплексу в її рамках.

2.3.2. Побудова моделі навчально-методичного комплексу, та його взаємодія із іншими компонентами

Перед тим, як перейти до розгляду моделі навчально-методичного комплексу, розглянемо узагальнену схему розподілу функціоналу у платформі автоматизації навчального процесу. На основі отриманої моделі виділимо основні властивості кожного окремого елементу розроблюваної інформаційної системи, щоб визначити основні складові навчально-методичного комплексу, у відповідності із раніше розглянутою структурою освітнього процесу у вищих навчальних закладах.

З урахуванням мікросервісної архітектури, і виокремлення ряду допоміжних елементів, розроблена модель платформи зображена на рис. 2.5.

У відповідності до отриманої моделі, можна побачити що основний робочий функціонал реалізовують 5 окремих системи, які виступають як окремі модулі, незалежні один від одного. Всі модулі взаємодіють із централізованою базою даних, в якій зберігаються всі необхідні дані для роботи як самої платформи, так і окремих її компонентів, які потребують інформацію із неї.

У відповідності до моделі, основні компоненти системи такі: Система інформаційної інфраструктури компонентів Університету, Адміністративна система Університету, Навчально-методичний комплекс, Система наукового забезпечення, Система забезпечення функціональних процесів. Розглянемо детальніше 4 основних компонента у відповідності до функцій, які вони реалізовують, і на основі них сформуємо функціонал, за яким буде створено 5 модуль – навчально-методичний комплекс.

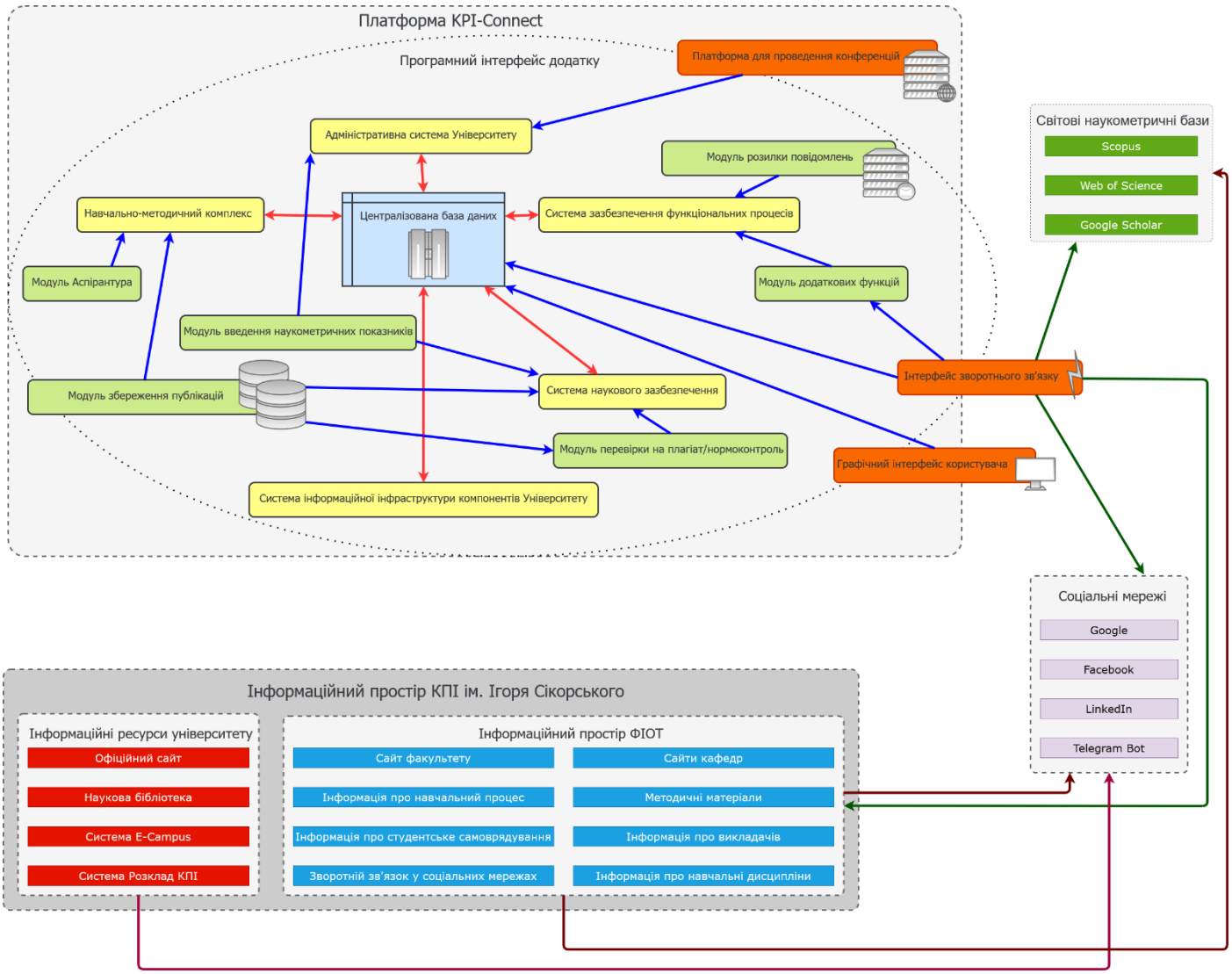


Рис. 2.5. Модель інформаційної платформи автоматизації навчального процесу

Система інформаційної інфраструктури компонентів Університету.

Даний модуль платформи призначена для формування загальної структури університету. Для цього, в основі цієї системи включені основні структурні компоненти, які моделюють логіку компонентів університету. В якості базового елементу виступає компонент Підрозділ. За допомогою відповідного поля Тип Підрозділу формується окремий запис у відповідності до його структури. Серед основних типів слід зазначити факультет, інститут, кафедра, деканат, ректорат, адміністрація, студентська рада, та ряд інших. Для побудови залежності одних структур від інших, кожний окремий запис може визначити батьківський підрозділ, який відповідає йому.

Крім зазначеної моделі структури університету, даний модуль також визначає базові структури компонентів освіти – Спеціальність, Спеціалізація, Напрямок навчання, Рівень вищої освіти, Форма навчання, Кваліфікація. Ряд цих компонентів використовується для побудови основи організації освітнього процесу, а саме – формування освітніх програм і навчальних планів. Їх взаємодія із сутностями Навчально-методичного комплексу буде розглянуто пізніше.

Адміністративна система Університету.

Основним компонентом роботи платформи виступає Адміністративна система Університету. Цей модуль включає реалізацію основної структури роботи із платформою – кадрові ресурси університету. В модулі вказані основні поля, з яких складається сутність Персона, та пов’язані із нею ключові ролі, з яких складаються процеси у навчальних закладах – Студент і Викладач.

Крім визначеної моделі кадрових ресурсів, система передбачає введення окремих сутностей взаємозв’язку кадрового складу із Напрямками роботи, визначеними за різними підрозділами ВНЗ, а також окремі посади, які відповідають як окремим кафедрам, так факультетам і інститутам в тому числі. В якості іншого важливого функціоналу цього модуля можна назвати побудова його взамозв’язку із іншими компонентами платформи, так як кожен окремий модуль частково або повністю залежить від значень із відповідних ролей кадрових ресурсів системи.

Слід також врахувати, що цей модель враховує функціонал збереження інформації про Освітній статус її користувачів на поточний момент. Він стосується як і Студентів, так і Викладачів-користувачів системи, відповідно.

Система наукового забезпечення.

Даний компонент платформи визначає елементи наукової діяльності, які пов’язані із навчальним процесом. В рамках цього модуля виокремлено сутності Наукові публікації, а також Наукові проєкти, які проводяться у різних підрозділах навчального закладу.

Для збереження даних пов’язаних із науковою складовою навчання, цей модуль використовує додаткову сутність Документ. Вона визначає зв’язок і інформацію про наданий файл із науковою публікацією, чи відповідним нормативним документом, який відповідає науковому проєкту, і зберігає у окремому файловому сховищі платформи.

Окремою сутністю в рамках цього модуля постає Дисертація. Вона визначає ключові поля і елементи щодо дисертацій, дипломних робіт і проєктів для студентів і викладачів системи, і визначає порядок надання нового рівню вищої освіти.

Система забезпечення функціональних процесів.

Даний компонент системи виконує узагальнену роль моделювання функціональних процесів, які відбуваються у вищому навчальному закладі. Структура таких процесів передбачає виконування деяких певних дій, які передбачають правильність виконання циклу навчання як студентів, так і викладачів.

Наприклад, типовий цикл проходження навчання студента передбачає наступні події: вступ у ВНЗ, проходження навчального процесу кожен семестр, із проходженням відповідних контрольних заходів (екзаменаційна сесія, атестація, ректорський контроль та інші), проєктування і написання дисертаційної роботи або здача підсумкових іспитів. Цей розділ платформи передбачає виконування зазначених елементів у відповідності із проходженням навчання у університеті, враховуючи усі підпроцеси, які можуть спіткати студентів під час навчання на різних семестрах навчання.

Слід також врахувати, що зазначена система є тісно пов’язаною із модулем додаткових функцій, який може розширюватися в залежності від необхідного функціоналу для виконання поточного навчального процесу в Університеті.

Навчально-методичний комплекс.

Як вже було розглянуто раніше, кожний окремий основний модуль підсистеми виконує ряд базових функцій, які допомагають сформувати процес навчання у відповідності із передбаченим для цього положенням. Навчально-методичний комплекс доповнює організацію навчального процесу у відповідності із основними компонентами навчання, і допомагає сформувати правильний контроль за поточною успішністю кожного студента-користувача платформи.

Використовуючи попередньо виконаний аналіз типових електронних навчально-методичних комплексів, у відповідності до сформованої моделі інформаційної платформи автоматизації навчального процесу, було сформовано наступну структуру даного модуля.

В основі роботи цього компоненту використовується сутність Навчального плану. Як другий елемент організації навчального процесу після освітньої програми, дана сутність визначає основні дисципліни, які будуть вестися за окремою спеціальністю і спеціалізацією для певної групи здобувачів освіти на весь період навчання. Для цього, буде організовано зв’язок із Системою інформаційної інфраструктури платформи, для формування необхідних полей навчальних планів для окремих спеціальностей у відповідних кафедрах університету.

Для введення контролю за навчанням у комплексі було виділено такі окремі елементи як Предмет і Оцінка. Предмет визначає ряд дисциплін, які в цілому викладаються на різних спеціальностях ВНЗ, а Оцінка виступатиме як фактор контрольних заходів для перевірки процесу навчання. Для цього, в цій сутності буде визначено окреме поле, яке зазначає тип контролю, який проводився відповідно до поточної навчальної діяльності, і викладач матиме змогу додати відповідне значення оцінки з поясненням. На основі даних із цих сутностей можна буде формувати поточну успішність за окремими групами чи вибіркою студентів, на основі яких формувати аналіз засвоєння інформації під час навчального процесу.

Для забезпечення методичної складової ЕНМК, у даний модуль буде додатково введено сутність Документ з необхідним функціоналом, який є аналогічним до такої, яка виступає елементом системи наукового забезпечення. Використання цього елементу дозволить надавати студентам посилання на необхідну навчально-методичну літературу, і за потреби давати змогу ознайомлюватися із науковим матеріалом викладачів і наукових співробітників, внесених у систему. Для правильної роботи цієї сутності модель передбачає зв’язок із Об’єктним сховищем платформи, який забезпечує доступ до всіх збережених файлів.

В результаті розглянутих складових даного комплексу, можна сказати, що його побудова із використанням компонентів загальної платформи автоматизації навчального процесу дозволяє спростити процес розробки окремих її функціональних елементів, і при цьому забезпечити організацію освітнього процесу у відповідності до розглянутого положення НТУУ «КПІ ім. Ігоря Сікорського». Інтеграція загальної платформи із різними світовими інформаційними ресурсами дозволить в тому числі навчально-методичному комплексу застосовувати інші окремі мережеві додатки для доповнення організації освітнього процесу, з метою покращення навчання для здобувачів освіти. Як приклад, такими додатковими ресурсами можуть виступати сервіси Google Apps, система Moodle відповідного підрозділу з інформаційного простору Факультету Інформатики та Обчислювальної техніки, тощо.

На основі розглянутих структурних компонентів, модель Навчально-методичного комплексу із використанням зв’язків із іншими основними системами проєкту зображено на рис. 2.6:

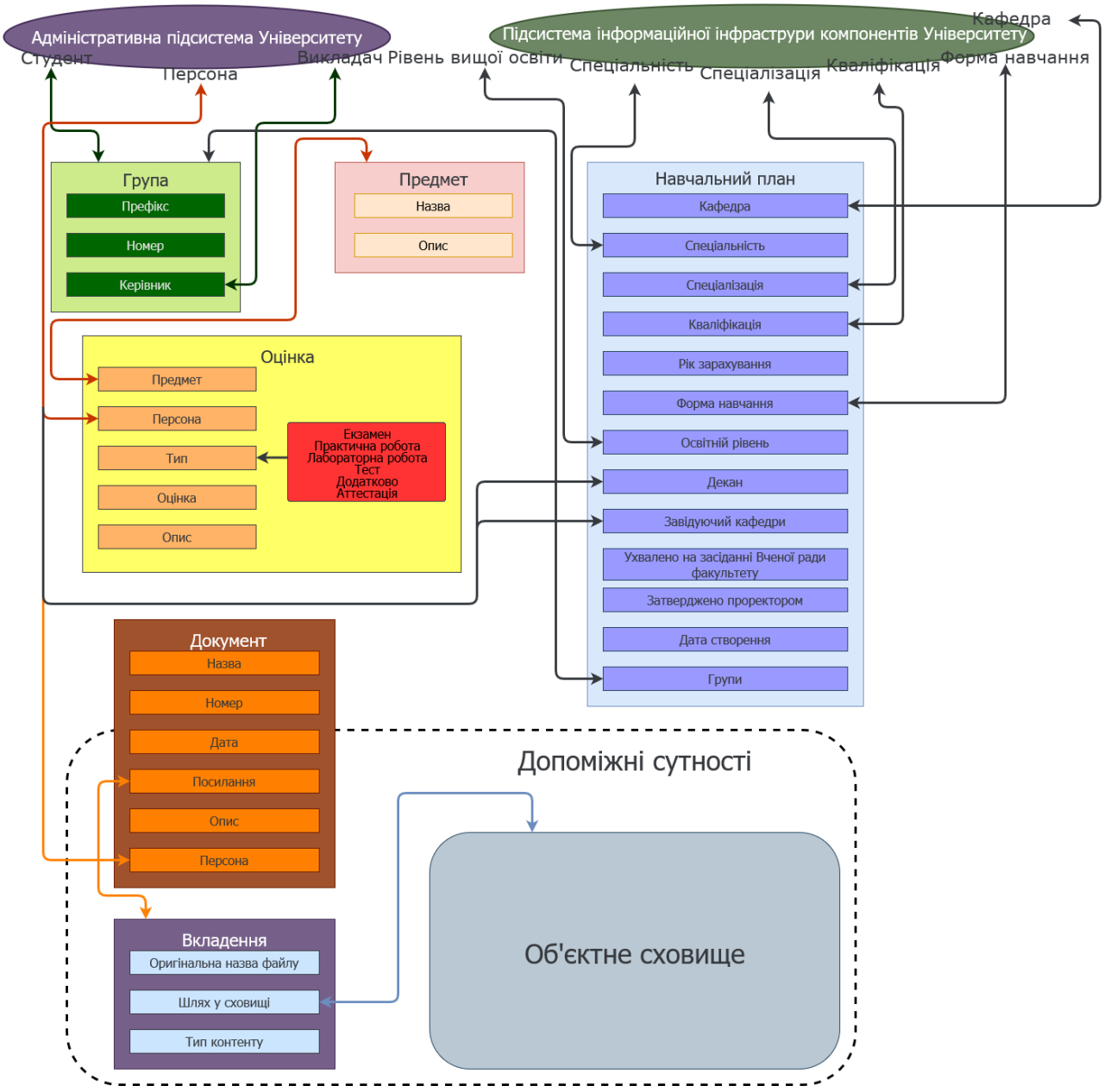


Рис. 2.6. Модель навчально-методичного комплексу із застосуванням зв’язків з іншими компонентами інформаційної платформи

Висновки до розділу 2

В результаті порівняльної характеристики розглянутих прикладів інформаційних систем у різних університетах, було визначено, що кожна окрема реалізація такої платформи передбачає різні можливості і функціонал. Кожен окремий університет обирає ряд необхідного функціоналу для реалізації деяких функцій покращення продуктивності навчальним процесом, і більша кількість додаткових функцій як правило визначається комерційним рішенням для побудови ефективної інформаційної системи.

Аналізуючи проблему подачі електронного навчально-методичного комплексу можна визначити велику кількість окремих елементів і структур, необхідних для ефективної побудови цільної системи контролю за навчальним процесом. Як правило, набір важливих процесів в рамках навчання також визначається окремими документальними положеннями в різних університетах, що в свою чергу потребує побудови структури системи університету в електронній формі.

На основі такого аналізу було прийнято реалізовувати окремий механізм навчально-методичного комплексу, який будується за постановою НТУУ «КПІ ім. Ігоря Сікорського» в рамках окремої інформаційної платформи автоматизації навчального процесу «KPI-Connect», яка у своїй мікросервісній архітектурі вже передбачає електронну складову університету, і у свою чергу є простою і потужною модульною системою розробленою власноруч.

В рамках такої платформи було представлено окрему модель компоненту навчально-методичного комплексу, який з’єднаний із центральною базою даних, і формує основу платформи. Вибір технологій реалізації цієї складової, а також опис окремих конфігураційних файлів, необхідних для побудови загальної платформи виконання коду, а також моніторингу стану системи і оцінки якості її роботи буде виконано в рамках Розділу 3 цієї роботи.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ОТРИМАНОЇ МОДЕЛІ НАВЧАЛЬНО-МЕТОДИЧНОГО КОМПЛЕКСУ

3.1. Вибір технології розробки

Розглянемо обрані технології розробки у відповідності до окремих складових системи. Для цього розділимо аналіз обраних технологій між апаратною складовою і власне програмними технологіями розробки коду роботи.

В якості основи роботи платформи і розподілу апаратних ресурсів для побудови мікросервісної архітектури, було застосовано технологію контейнеризації окремих складових платформи. Для цього було обрано програмний продукт Docker, та було описано відповідні образи окремих складових компонентів. Перейдемо до детального огляду обраної складової.

3.1.1. Опис апаратної складової платформи у вигляді контейнерів Docker

Система Docker представляє у собі програмний додаток, що виконує у собі роль управління контейнерами інших програмних засобів. Він працює таким чином, що дозволяє «запаковувати» різні програмні рішення-сервіси із усім його пов’язаним середовищем і залежностями у деякий віртуальний контейнер, який можна запускати і управляти на окремій операційній системі Linux за допомогою наданого набору команд. Виходячи із такого визначення, Docker представляє собою засіб віртуалізації програмних засобів у комп’ютерних системах на рівні операційної системи.

Вибір даного типу платформи будується на його популярності і простоті реалізації внутрішньої архітектури програмного забезпечення. Docker застосовуючи технологію контейнерів надає можливість самій системі управляти необхідними ресурсами для керування внутрішніх додатків і сервісів, які працюють в них. Крім того, застосовуючи модель заданого інструментарію

програми, існує можливість просто масштабувати програмні додатки горизонтально за допомогою розгортання додаткових контейнерів на окремих Docker хостах для вирішення поставленої задачі [26]. Популярність цього засобу можна побачити у відповідності до рис. 3.1., на якому зображено кількість пошукових запитів Docker у порівнянні з іншими засобами – віртуалізацією від компанії VMware, Hyper-V, KVM, а також інший відомий засіб віртуалізації Kubernetes:

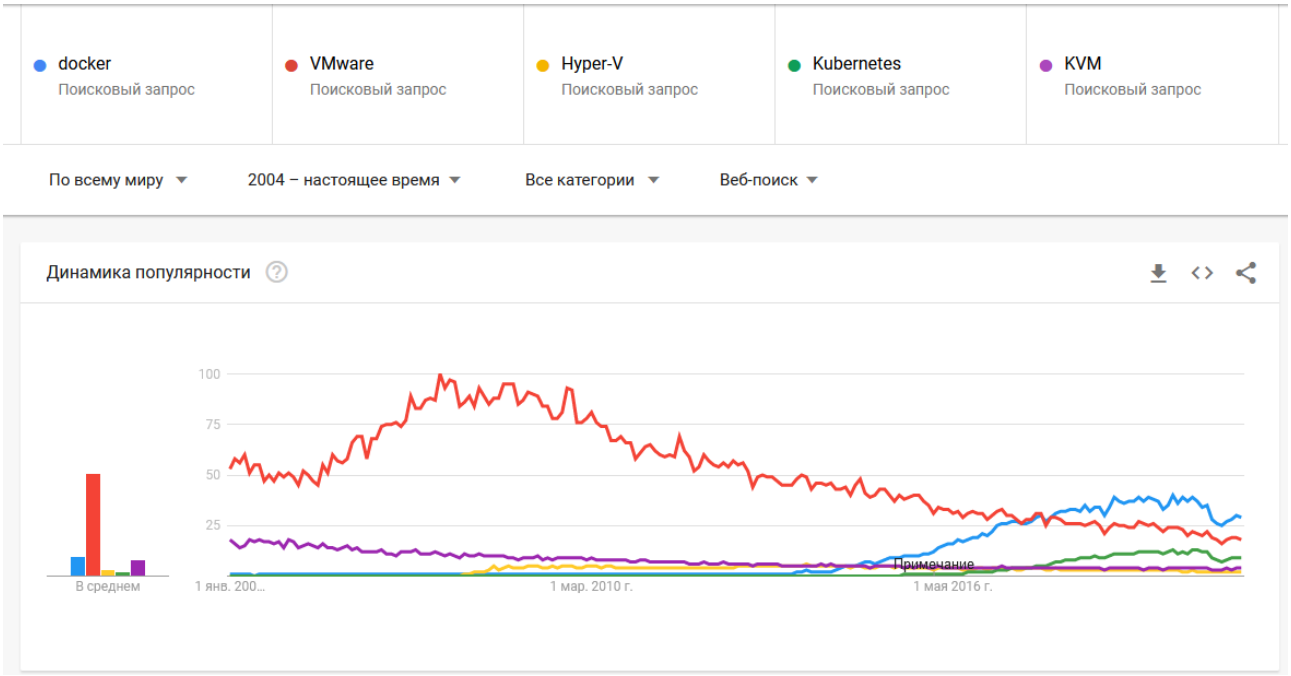


Рис. 3.1. Динаміка популярності засобів віртуалізації у порівнянні з Docker [29]

Існують декілька важливих компонентів, на основі якого формується система Docker – образ (зображення), реєстр і власне контейнер. Розглянемо кожен із цих термінів окремо:

- Образ (зображення) – деякий шаблон, який представляє у собі елемент для створення і роботи самих контейнерів. Найчастіше він має вигляд у формі копії файлової системи із програмними засобами відповідного сервісу з його оточенням;
- Реєстр – репозиторій зображень. Існують як і локальний реєстр зображень, збережених на локальному комп'ютері, так і віддалені публічні реєстри, на яких зберігається велика кількість публічних

зображень, доступних для використання в якості розробки або тестування;

- Контейнер – запущений додаток на хості Docker, який представляє у собі сукупність працюючих процесів, і файлової системи зображення.

Для взаємодії із набором інструментів, які надає сама система Docker, існують як окремий програмний інтерфейс (API), так і командний інтерфейс через термінал (CLI). Крім того, для організації взаємодії між контейнерами, інструментарій надає можливість створювати віртуалізовану мережу, із відповідною конфігурацією мережі на рівні самої операційної системи. А також, Docker надає можливість створення віртуальних топів – окремих дисків для контейнерів, з якими вони можуть працювати окремо від ядра самої системи, а також використовуватися в якості взаємного сховища спільних даних [24].

Якщо для розробки додатку або інформаційної системи необхідно застосування декількох окремих контейнерів для відділення сервісів один від одного, або застосування окремих образів для кожного елементу, від розробників компанії надається додаткове рішення, що має назву Docker Compose. Для побудови правильного кластеру контейнерів, Docker Compose надає можливість створення окремого конфігураційного файлу у форматі YAML, в якому описуються всі складові кожного контейнеру, їх типи образів, та застосовані віртуальні мережі для комунікації і томи для збереження даних.

Функціонал Docker надає також можливість створення окремих власних образів систем, які можуть працювати у контейнері на основі існуючих. Побудова таких контейнерів використовуючи систему Docker Compose передбачає застосування додаткового конфігураційного файлу, що має назву Dockerfile. В рамках цього файлу виконується опис складових зображення, яке створюється на базі існуючого, а також набір команд необхідних для правильної підготовки початкового стану контейнеру перед його запуском у Docker [25].

Надалі будуть розглядатися технології розробки, використані для побудови коду програмного продукту, а також ряд засобів, які використовувалися для тестування роботи програмного комплексу і аналізу застосування ресурсів.

Розглядаючи обрану мікросервісну архітектуру додатку, зазначені засоби будуть оглядатися в рамках окремих контейнерів Docker у відповідності до їх функціоналу.

3.1.2. Опис програмних засобів, застосованих при розробці коду навчально-методичного комплексу

Функціонал програмної складової окремо заданого модуля базується на backend та frontend складових, необхідних для відображення контенту веб-сторінок самого комплексу. Розглянемо ці складові окремо один від одного.

Backend складова розробленого комплексу базується на *мові програмування PHP*. Це є популярною мовою для написання бізнес логіки роботи процесів у деякому веб застосунку, і має ряд додаткових модулів, які дозволяють спростити процес написання коду, та формування загальної структури проєкту розробки. В даному елементів якості формування структури *MVC* (Model-View-Controller, Модель-Представлення-Контролер) моделі застосунку використовується *фреймворк Symfony*.

Даний інструмент розробки надає просту модель побудови деякого веб-додатку, із можливістю легко отримувати необхідну інформацію про окремі важливі дані із реляційних баз даних, застосовуючи структуру об’єктної моделі. Symfony має підтримку великої кількості сучасних баз даних, які підтримують модель *PHP Data Objects* (PDO). Для роботи із ними Symfony застосовує *інструмент ORM* (Object Relation Manager, об’єктно-реляційний проектор), і в даній системі було використано популярний засіб для цього фреймворку *Doctrine*.

Даний фреймворк також передбачає можливість застосування додаткових пакетів, які можуть надавати ряд додаткових засобів із побудови структурованого програмного коду. Так, для організації адміністраторської панелі усієї платформи було використано *пакет SonataAdminBundle*. Цей пакет надає ряд доступних класів розробки із фреймворком Symfony, і дозволяє побудувати простий і зручний інтерфейс доступу адміністраторам платформи до

основних компонентів, з яких вона складається. Для цього, цей окремий пакет має ряд вбудованого функціоналу, який із використанням об'єктної моделі Doctrine має змогу отримувати усі необхідні дані із підключеної бази даних веб-застосунку, і формувати відповідну інформацію.

Крім того, в рамках додаткових пакетів для Symfony було застосовано окремий пакет для побудови зовнішнього інтерфейсу аутентифікації користувачів. Він має назву *FOSUserBundle*. Даний пакет надає додатковий функціонал розподілу ролей користувачів у платформі, дозволяє побудувати систему входу і реєстрації у користувацький інтерфейс. Кожен користувач у відповідності до моделі ролей з цього пакету дозволяє надавати йому ряд прав, які дозволяють переходити у зазначені для них розроблені форми інтерфейсу, із можливістю виконувати тільки ті задачі, які передбачені для цієї ролі. Слід зазначити, як частина цього пакету виступає також окремий функціонал *бібліотеки OAuth*, що дозволяє реєструватися і входити у систему користувачам за допомогою популярних соціальних мереж і сервісів – Google, Linkedin, Facebook. Уся необхідна для входу інформація із зазначених соціальних мереж зберігається у базі даних таблиці користувачів, що дозволить користувачам в подальшому застосовувати цю інформацію для входу.

Для розробки frontend складової всієї інформаційної платформи автоматизації навчального процесу було застосовано *підхід SPA* (Single Page Application, односторінковий застосунок), який передбачає, що уся робота користувача із інтерфейсом системи не передбачає переходів на інші веб сторінки системи. При такій моделі застосовується динамічний зв'язок між клієнтом і сервером, при якому надходження додаткових елементів сторінки відбувається у реальному часі, у відповідь на дії користувачів.

Для реалізації цього підходу було обрано *мову програмування JavaScript*, яка надає можливість завантаження інформації про веб додаток без жодних затримок, у відповідь на активний зв'язок із самим веб-сервером. Для спрощення побудови цього процесу було використано також *фреймворк React*, що надає можливості до динамічних змін з обробки інформації на поточній веб-сторінці, і

таким чином найкраще підходить для виконання поставленої задачі односторінкового додатку.

Крім використання зазначеного фреймворку важливим елементом роботи SPA має виступати елемент збереження поточного стану інтерфейсу платформи. Для правильного застосування цього компоненту було використано окрему бібліотеку, яка працює за допомогою структур React, і має назву *Redux*. Ця бібліотека надає можливість управляти зручно станами певних компонентів інтерфейсу, відокремити доступ до відповідного сховища даних стану, і оперувати ним в залежності від потреби.

Для спрощення процесу розробки окремих компонентів інтерфейсу користувача було вирішено застосовувати окрему бібліотеку, яка надає стандартизований дизайн основних компонентів роботи із веб-додатками. Досить популярною *бібліотекою* вважаться *Material UI*, яка надає набір компонентів із такою структурою, яка дозволяє максимально інтуїтивно просто користуватися інтерфейсом веб-застосунку, не перевантажуючи її важкими елементами. Так як її реалізація наявна і для фреймворку React, було вирішено і його застосувати в якості елементу основи роботи.

Зазначені технології розробки програмного коду формують основний функціонал окремого модуля навчально-методичного комплексу. Але, крім нього в якості реалізації повної структури проєкту, а також вибору окремих компонентів необхідних для проведення тестування програмного коду і оцінки якості роботи системи в цілому, було введено окремі сервіси моніторингу, реалізовані також у вигляді окремих контейнерів Docker. Розглянемо їх також детальніше у наступному під модулі.

3.1.3. Опис додаткових сервісів реалізації тестування і моніторингу роботи системи

Крім технологій розробки програмного коду слід врахувати окремі сервіси, які також приймають важливу участь у роботі всієї платформи в цілому. Для

повноцінної роботи як і платформи, так і навчально-методичного комплексу слід зазначити такі важливі елементи роботи як веб-сервер і центральна база даних.

Як окремий контейнер системи було застосовано модуль веб-серверу, який видає оброблену інформацію із компоненту бізнес логіки окремих складових модулів, і надає інформацію користувачам, очікуючи їх відповіді. В даній моделі було вибрано для цього веб-сервер *nginx,* через його ряд переваг при розробці веб-додатків. У порівнянні з іншим популярним сервером Apache, nginx є легковажним, і реалізовує масштабованість набагато краще. Крім того, він досить просто може бути налагодженим, і готовим для виконання своєї роботи.

В якості централізованої бази даних було вибрано модель із реляційним підходом, *СУБД* (Система управління базами даних) *PostgreSQL*. Даний вид бази даних є одних із найпопулярніших типів, які обираються для розробки веб-додатків, і має також ряд своїх переваг. Серед плюсів цієї СУБД можна назвати підтримку механізмів реплікації і транзакцій, а також її легку масштабованість. Крім того, дана база даних частково підтримує об’єктні структури даних, що дозволяє зберігати у комірках інформацію, яка складається з комплексних складових, наприклад масиви.

Як один з допоміжних контейнерів для роботи навчально-методичного комплексу було застосовано контейнер для *об’єктного сховища*. Як окрема структура, він передбачає систему збереження об’єктних даних, такі як файли методичних даних, наукові роботи і дисертації, тощо із застосуванням технології S3. В рамках даної роботи було використано готовий образ такого сховища у вигляді додатку *MinIO*.

Крім зазначених основних компонентів роботи системи, у загальну архітектуру платформи було також введено ряд окремих компонентів, які реалізовують тестування роботи системи, перевірка якості програмного коду, а також аналіз стану у вигляді комплексного моніторингу. Розглянемо технології, використані для реалізації цих компонентів.

В якості платформи збору інформації і моніторингу стану роботи системи в цілому було застосовано окремий контейнер із програмним засобом *Prometheus*.

Ця програма виступає як продукт з відкритим кодом, яка збирає метрики роботи платформи в цілому з різних її програмних компонентів, апаратних складових, робочий стан тої чи іншої функції окремого модуля, тощо. Принцип роботи базується на основі моделі HTTP запитів у об'єктів дослідження, і отримання відповідних метрик у реальному часі. Керуванням кількістю запитів для збору даних може бути легко налагоджено через відповідний конфігураційний файл prometheus.yml, детальніше на прикладі розглянемо його пізніше.

Для збору метрик із різних елементів платформи, моніторинг система передбачає ряд різних засобів і окремих модулів, які дозволяють генерувати деякі показники різних компонентів платформи, і надсилати на постійний аналіз у центральний сервер Prometheus. Для збору даних застосовуються три різні підходи:

- Інструментування – налагодження додатку платформи відповідними бібліотеками програмування для сформування власних метрик, які може розпізнавати Prometheus. Дані бібліотеки при їх впровадженні у програмний код формують власні об'єкти пам'яті у вигляді деяких лічильників або вимірювачів, які самі можуть динамічно змінювати значення, і на основі них формувати метрики роботи програми. На поточний момент існує ряд таких бібліотек для ряду популярних мов програмування (Python, Java, Go, Ruby), які офіційно надаються самим проєктом Prometheus. Крім того, існує ряд сторонніх бібліотек від користувачів для інших мов (C++, C#, NodeJS, PHP), які так само надають можливість вбудовувати системи метрик у власні програми.
- Експортери – окремі сервіси для відомих додатків і засобів програмування, які мають вже деякий набір метрик і об'єктів, за якими можна моніторити стан системи на деякому прикладі. Серед відомих таких сервісів можна назвати розробки експортерів для баз даних, враховуючи усі популярні СУБД на поточний час; експортери HTTP, які застосовуються для веб-серверів; експортери Unix та власне контейнерів

Docker, які виконують моніторинг ключових вузлів самої операційної системи, та стан контейнерів, на яких працює додаток.

- *Pushgateway* – використання окремого сервісу, який збирає метрики із додатку самостійно. Даний підхід застосовується при умовах, коли розроблюване програмне забезпечення не можуть надавати метрики напряму на сервер Prometheus, або тип метрик не передбачає їх надсилання у звичайному режимі опитування модулів інформаційної системи. Прикладом таких метрик можна назвати пакетні завдання, інформацію про яких моніторити можна у відповідності до обраних задач [20].

Приклад того, як Prometheus збирає метрики із відповідних вузлів деякої платформи показано на рис. 3.2:

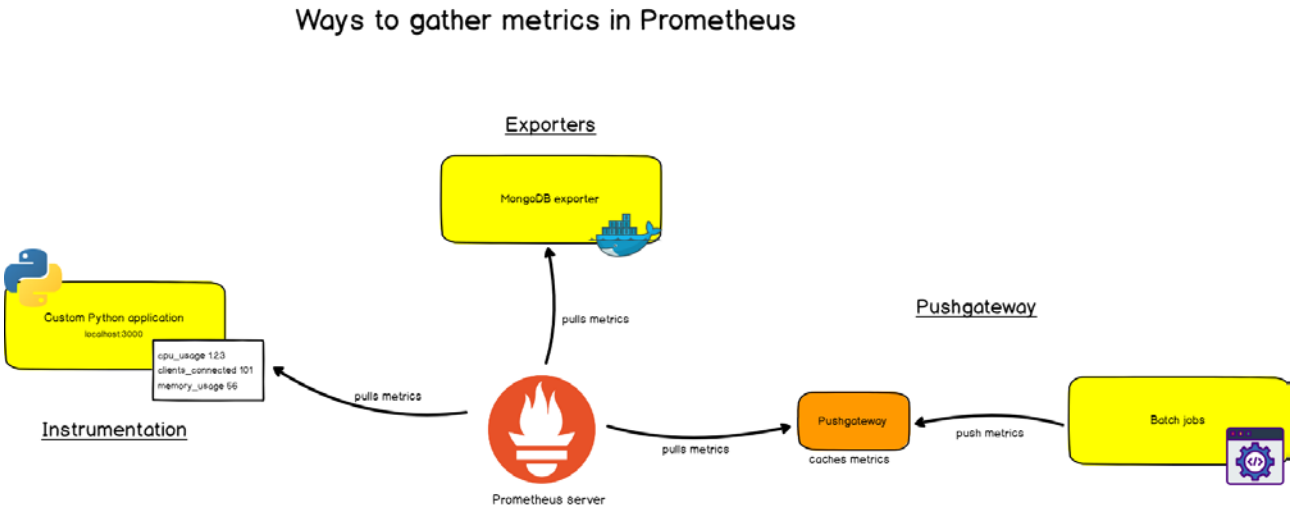


Рис. 3.2. Способи збору метрик із об’єктів дослідження, використовуючи сервіс Prometheus [34]

У відповідності до виконуваної роботи, для збору і аналізу стану системи в цілому було вирішено виділити наступні засоби, які будуть використовуватися для збору даних у нашій платформі:

1. Модуль *node-exporter*. Даний експортер призначений для збору метрик і даних з апаратних складових і системних для операційних систем типу Unix. Так як контейнери Docker працюють на основі хост машини з операційною системою Linux, застосування даного експортера є важливим критерієм перевірки стану роботи самого хоста, та його

складових, таких як оперативна пам’ять, процесор, жорсткий диск, та ключовий стан основних сервісів роботи самої ОС.

2. Компонент *cAdvisor*. Даний елемент надає набір інструментів та виділених метрик для перевірки стану роботи власне контейнерів. Він має власну підтримку роботи із двигуном Docker, та надає набір ключових метрик для системи Prometheus у початковому вигляді.
3. Сервіс *PostgreSQL Server Exporter*. Виходячи із назви цього сервісу, цей експортер виконує роль збору даних зі стану роботи СУБД типу PostgreSQL. Цей модуль використовується для оцінки робочого стану основної бази даних, щоб проаналізувати її продуктивність роботи.
4. Модуль *PHP-FPM Exporter*. Окремий експортер для збору метрик і даних із інтерфейсу програмного коду PHP FastCGI, який застосовується для збереження програмного коду платформи. Використання цього модулю надає можливість перевіряти стан роботу процесору обробки PHP, який реалізовується в окремих сервісах системи, і надавати дані для моніторингу у сервер Prometheus.
5. Додатковий пакет до фреймворку Symfony – *Prometheus Metrics Bundle*. Цей пакет надає функціональні можливості збирання необхідних метрик із роботи власне веб-додатку, і постачається із готовим клієнтом інструментування Prometheus для мови програмування PHP – *Jimdo/Prometheus_client_php*. Налагодження пакету відбувається в рамках конфігураційних файлів самого проєкту додатку.
6. Додатковий пакет до SonarQube – *SonarQube Prometheus Exporter*. Даний експортер надає можливість збирати метрики із застосунку перевірки якості коду SonarQube у систему Prometheus, та використовувати їх для моніторингу стану розробки системи.

Також, окрім збору і збереження метрик Prometheus поставляється із додатковим пакетом, який може генерувати відповідні події та попередження, у відповідності до значення поточних метрик. Такий пакет має назву Alert Manager, і він налагоджується як окремий сервіс поряд із самим сервером

Prometheus. Управління типом повідомлень на основі самих метрик виконується на рівні Prometheus, в той же час відповідний сервіс відповідальний за правила відправки у різноманітні сервіси, які включають електронну пошту, бізнес додаток Slack, або у сервіс реагування на інциденти PagerDuty.

Крім збору метрик, і зберігання їх у базі даних серверу Prometheus у вигляді файлів, необхідно побудувати додатковий інтерфейс моніторингу стану метрик. Для цього, існує окремий *сервіс візуалізації даних метрик*, що має назву *Grafana*. Prometheus надає доступ до даних результатів метрик використовуючи власний синтаксис PromQL, який може бути застосований для побудови статистики зібраної інформації. Grafana маючи підтримку з обробки даних запитів PromQL надає можливість побудувати власну панель із графіків і діаграм, які були зібрані у відповідних метриках сервісів платформи. Конфігурація панелі моніторингу відбувається через окремий графічний інтерфейс, або через виконання відповідних змін у файлі типу .json. Побудова великої кількості моніторингової інформації через відповідні окремі сервіси платформи збільшує розмір файлу json, що у свою чергу призводить до конфігурації панелі через графічний інтерфейс додатку у більш простіший спосіб.

Крім зазначених засобів моніторингу стану роботи системи та її компонентів в рамках платформи також реалізовано використання сервісу перевірки стану якості програмного коду додатку. Для цієї задачі було застосовано *застосунок SonarQube*. Даний додаток надає змогу аналізувати якість написання деякого програмного коду. В основі своєї роботи SonarQube бере деякий знімок стану коду на поточний момент часу, перевіряє відповідність роботи деякому стандарту із можливим місцем знаходження потенційних помилок чи навіть багів розробки, і повідомляє в якості результату відсоткове значення якості написаного коду із відповідною кількістю попереджень чи помилок. Використання моделі знімків у часі розробник додатку може слідкувати за якістю зміни написання коду, і на основі цих результатів робити аналіз чи покращується стандарт написання програмного забезпечення.

Ця система перевірки написання коду розміщується схожим чином, як і моніторинговий додаток Prometheus – необхідно розмістити окремий сервер для збереження даних перевірки роботи за проєктами, а для перевірки коду у самому додатку розміщується окремий пакет сканеру коду, який і виконує саму перевірку. Зв’язок між сканерами і основним сервером SonarQube виконується за допомогою протоколу HTTP. Для збереження результатів перевірки, і генерації отриманих даних сервер SonarQube потребує встановлення окремої бази даних, яка також буде працювати як окремий сервіс поряд із самим сервером. Для доступу до основного вікна конфігурації програми достатньо перейти на URL сторінку самого серверу SonarQube, де виконується налагодження його роботи, може бути застосовано адміністрування доступу роботи до веб-додатку у відповідності із правами доступу, а також існує можливість переглядати поточний стан перевірки коду розроблюваного застосунку, і історію перевірок [23].

В розроблюваній платформі результати перевірки якості коду будуть відправлятися у моніторингову систему за допомогою відповідного експортеру SonarQube Prometheus Exporter. Перегляд результатів як окремих метрик самого Prometheus буде виконуватися через відповідну панель у сервісі Grafana.

У відповідності до розглянутих усіх компонентів розробки і розгортування моніторингової системи перевірки метрик роботи усіх складових системи, а також додатку перевірки якості написання коду, перейдемо до опису конфігураційних файлів для побудови платформи.

3.2. Створення конфігурації роботи розроблюваної платформи

Як вже було розглянуто раніше, в якості апаратної складової роботи кожного компоненту платформи, було використано контейнеризацію за допомогою двигуна Docker, яке дозволяє побудувати всю модель системи на основі мікросервісної архітектури. Для побудови правильної структури кожного окремого контейнеру було застосовано програмне рішення Docker-Compose, яке

дозволяє згрупувати роботу деяких логічно визначених контейнерів в рамках одного конфігураційного файлу.

3.2.1. Конфігураційні файли docker-compose.yml

У відповідності до окремих розглянутих ключових компонентів, реалізація платформи для побудови навчально-методичного комплексу слід розділити на 3 окремих конфігураційних файли Docker Compose, у відповідності до окремих хостів контейнеризації Docker, які будуть застосовані для роботи самого продукту:

1. Перша конфігурація сервісів передбачає набір сервісів, необхідних для роботи власне розроблюваного додатку навчально-методичного комплексу, його основних складових, а також додаткових сервісів моніторингу стану компонентів і перевірки якості написання коду.
2. Друга конфігурація передбачає розгортання серверу моніторингу стану компонентів Prometheus, а також засобу візуалізації отриманих метрик на основі нього Grafana;
3. Третя конфігурація необхідна для побудови окремого серверу перевірки якості коду SonarQube, відповідного його СУБД PostgreSQL для збереження результатів перевірки, а також додаткові сервіси-експортери для збирання метрик станів компонентів.

Опис першого конфігураційного файлу docker-compose.yml показано на рис. 3.3-3.5. Розглянемо детальніше його ключові складові, які описують конфігурацію окремих контейнерів.

Version: ‘2’ – конфігурація, яка описує версію Docker-Compose, який використовується для створення контейнерів. 2 версія показує другий стандарт роботи із контейнерами, і в основному передбачає роботу із контейнерами в рамках одного хосту Docker.

Services – в рамках цього поля описано детально окремі контейнери Docker, які будуть управлятися за допомогою Docker-Compose.

```yaml
version: '2'
services:
  kpi_connect_app_nginx:
    build:
      context: containers/nginx
      dockerfile: ./Dockerfile
    ports:
      - 80:80
    volumes:
      - ../../:/var/kpi-connect:cached
    depends_on:
      - kpi_connect_app_fpm
    networks:
      - kpi_connect_app_network

  kpi_connect_app_postgresql:
    image: postgres
    ports:
      - 5432:5432
    volumes:
      - ../../var/postgresql/db:/var/lib/postgresql
    environment:
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=admin
      - POSTGRES_DB=kpi_connect
    networks:
      - kpi_connect_app_network

  kpi_connect_app_fpm:
    build:
      context: containers/fpm
      dockerfile: ./Dockerfile
    volumes:
      - ../../:/var/kpi-connect:cached
    expose:
      - 9001
    environment:
      PHP_XDEBUG_ENABLED: 1
      XDEBUG_CONFIG: remote_host=10.254.254.254
    depends_on:
      - kpi_connect_app_postgresql
    networks:
      - kpi_connect_app_network

  kpi_connect_app_minio:
    image: minio/minio:RELEASE.2019-04-23T23-50-36Z
    volumes:
      - ../../var/s3:/data
    ports:
      - 9002:9000
    environment:
      MINIO_ACCESS_KEY: user
      MINIO_SECRET_KEY: user
```

Рис. 3.3. Частина першого конфігураційного файлу docker-compose.yml

```
    networks:
      - kpi_connect_app_network
    command: server /data/minio/

  kpi_connect_app_node_exporter:
    image: prom/node-exporter:latest
    command:
      - '--path.rootfs=/host'
    networks:
      - kpi_connect_app_network
      - kpi_connect_overlay_network
    ports:
      - 9100:9100

  kpi_connect_app_cadvisor:
    image: google/cadvisor:latest
    volumes:
      - /:/rootfs:ro
      - /var/run:/var/run:rw
      - /sys/:sys:ro
      - /var/lib/docker:/var/lib/docker:ro
    networks:
      - kpi_connect_app_network
      - kpi_connect_overlay_network
    ports:
      - 9080:9080

  kpi_connect_app_fpm_exporter:
    image: hipages/php-fpm_exporter:latest
    depends_on:
      - kpi_connect_app_fpm
    environment:
      PHP_FPM_SCRAPE_URI: "tcp://kpi_connect_app_fpm:9001/status"
      PHP_FPM_LOG_LEVEL: "debug"
    networks:
      - kpi_connect_app_network
      - kpi_connect_overlay_network
    ports:
      - 9153: 9153

  kpi_connect_app_postgresql_exporter:
    image: wrouesnel/postgres_exporter:latest
    depends_on:
      - kpi_connect_app_postgresql
    environment:

      DATA_SOURCE_NAME="postgresql://kpi_connect:secret@kpi_connect_app_postgresql:5432/kpi_connect?sslm
      ode=disable"
    networks:
      - kpi_connect_app_network
      - kpi_connect_overlay_network
```

Рис. 3.4. Частина першого конфігураційного файлу docker-compose.yml

```
    ports:
      - 9432: 9432

  kpi_connect_app_sonar_scanner:
    image: sonarsource/sonar-scanner-cli:latest
    depends_on:
      - kpi_connect_app_fpm
    environment:
      - SONAR_HOST_URL="http://kpi_connect_sonar_sonarqube:9010"
      - SONAR_LOGIN=MYAPIKEY
      - SONAR_PROJECT=kpi-connect
    volumes:
      - ../../:/usr/src:cached
    user: "${SONAR_UID}:${SONAR_GID}"
    networks:
      - kpi_connect_app_network
      - kpi_connect_overlay_network

networks:
  kpi_connect_app_network:
    driver: bridge
    ipam:
     config:
       - subnet: 172.77.10.0/24

  kpi_connect_overlay_network:
    driver: overlay
    ipam:
     config:
       - subnet: 172.78.10.0/24
```

Рис. 3.5. Частина першого конфігураційного файлу docker-compose.yml

Кожний окремий сервіс, або контейнер було записано у відповідності до назви проєкту, а також образу, який буде виконуватися у ньому. Наприклад, kpi_connect_app_fpm визначає контейнер для PHP FastCGI інтерфейсу, в рамках якого описано бізнес логіку платформи і зберігається програмний код веб-додатку.

У відповідності до першого конфігураційного файлу, було визначено, що основними робочими контейнерами, які працюють на першому хості Docker є веб-сервер nginx, основна СУБД для платформи PostgreSQL, інтерфейс FastCGI PHP-FPM для роботи з веб-сервером, середовище для об'єктного сховища MinIO, експортери node_exporter, cAdvisor, PHP-FPM exporter, PostgreSQL exporter і сканер коду Sonar Scanner. Окремі образи для контейнерів побудовані на основі готових файлів із відкритого репозиторію Docker Hub, коли в той же час веб-сервер nginx, та інтерфейс для додатку PHP-FPM визначені додатковими окремими файлами Dockerfile, із набором додаткової конфігурації пакетів Linux.

Для визначення образу, який буде застосовано у контейнері, застосовується поле *image*. В той же час, для збірки образу контейнеру на основі Dockerfile, передбачається додаткове поле *build*, в якому зазначається контекст збірки *context*, та відповідний файл, на основі якого виконується збірка *dockerfile*.

Поле *environment* визначає набір додаткових змінних середовища, які необхідні для роботи окремого контейнеру. Так, наприклад, змінні *POSTGRES_USER, POSTGRES_PASSWORD, POSGRES_DB* для контейнеру із основною базою даних визначають конфігурацію користувача за замовчуванням, і основну базу даних веб-додатку, яка створюється при першій ініціалізації контейнеру.

Поле *volumes* використовується для підключення додаткових точок доступу до файлів із файлової системи хоста Docker. В рамках цього поля записується список файлів або папок, які будуть підключені до контейнеру, і перезаписані файли за замовчуванням, які знаходяться там з оригінального образу системи.

Networks – зазначає список віртуальних мереж Docker, до яких буде підключений заданий контейнер. Ці мережі можуть бути створені перед

побудовою контейнерів через Docker-Compose, або бути зазначені у самому файлі. У даному прикладі, за конфігурацією хостів Docker буде застосовано окремі приватні мережі в рамках кожного хоста із свою підмережею IP адрес (kpi_connect_app_network, kpi_connect_monitoring_network, kpi_connect_sonar_network), а також буде виділено мережу, яка поєднує окремі контейнери на різних хостах (kpi_connect_overlay_network). Відповідно, приватні мережі в рамках контейнерів одного хоста мають мережевий драйвер driver тип bridge, що надає окремим контейнерам доступ один до одного, а також з'єднуються із мережевою конфігурацією самого хоста Docker. Для мережі зв'язку між хостами застосовується тип драйверу overlay, який надає можливість з'єднання окремих контейнерів на різних хостах один з одним.

Ports – зазначає список портів, які контейнер може виставляти для доступу. В рамках цього поля зазначається список таких портів, а також їх відповідність порту хоста, на який має транслюватися доступ до контейнеру. В рамках конфігураційного файлу також зазначається схоже поле *expose*. Воно також використовується для зазначення виставлених портів, але таких які будуть доступні лише для внутрішніх сервісів.

Depends_on – поле, яке зазначає залежність деякого контейнеру від іншого. За допомогою цього поля можна визначити черговість запуску окремих контейнерів в рамках Docker-Compose, так як робота деяких сервісів неодмінно може залежати від вже запущених інших модулів. Як правило, першим сервісом для веб-додатків мають бути відповідні СУБД, щоб бізнес логіка мала можливість завантажувати необхідні дані з вже працюючого модуля.

Command – поле, яке переписує деяку команду за замовчуванням в рамках деякого контейнеру. Інколи, це необхідно для правильної ініціалізації і виконання додатку, який визначений у цьому контейнері.

У відповідності до розглянутих команд на прикладі першого конфігураційного файлу було розроблено таку ж структуру для другого і третього хостів Docker. Відповідні зображення їх конфігурації показано на рис. 3.6-3.8.

Другий конфігураційний файл docker-compose.yml виконує опис контейнерів для моніторингу роботи платформи. Він складається з образів додатку моніторингу Prometheus, системи візуалізації даних метрик Grafana, контейнеру для побудови системи сповіщень Alert Manager, а також експортеру стану хосту node-exporter.

Третій конфігураційний файл складається із контейнерів для перевірки якості написання програмного коду. Відповідні образи, які застосовані в ньому є такі: База даних для роботи системи перевірки якості коду PostgreSQL, власне система збору якості коду SonarQube, експортер перевірки стану хосту node-exporter і аналізу стану бази даних PostgreSQL exporter.

Для розгортання контейнерів необхідно кожний конфігураційний файл завантажити на відповідний хост Docker, із набором папок для кожного окремого сервісу. Запуск контейнерів виконується за допомогою команди `docker-compose up -d`.

Організація правильної роботи overlay мережі між контейнерами на різних хостах можлива якщо побудувати кластерний зв’язок між хостами. Для цього необхідно скористатися сервісом кластеризації декількох хостів Docker Swarm. Побудова кластеру за допомогою засобів Docker Swarm виконується наступним чином.

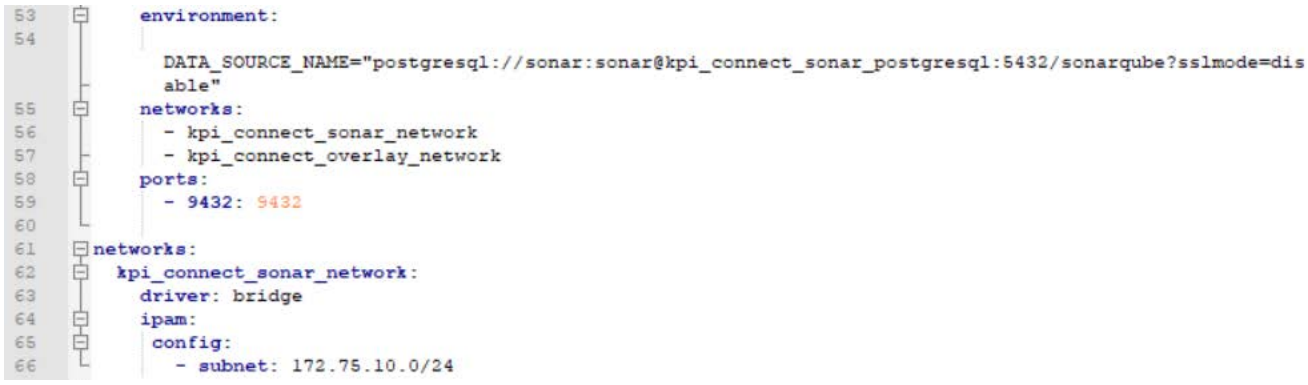
1. Спочатку основний хост створює роботу кластеру, за допомогою команди `docker swarm init`;
2. Усі інші хости мають змогу приєднатися до кластеру в якості робочих вузлів використовуючи команду `docker swarm join --token <your_token> <your_ip_address>:2377`.

```yaml
version: '2'
services:

  kpi_connect_monitoring_prometheus:
    image: prom/prometheus:latest
    volumes:
      - ./prometheus/config:/etc/prometheus/
      - ./prometheus/alerts/:/etc/prometheus/rules.d/
    command: --config.file=/etc/prometheus/prometheus.yml
    networks:
      - kpi_connect_monitoring_network
      - kpi-connect_overlay_network
    ports:
      - 9090:9090

  kpi_connect_monitoring_grafana:
    image: grafana/grafana:latest
    volumes:
      - ./grafana/provisioning/dashboards:/etc/grafana/provisioning/dashboards
      - ./grafana/provisioning/datasources:/etc/grafana/provisioning/datasources
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=password
      - GF_USERS_ALLOW_SIGN_UP=false
    networks:
      - kpi_connect_monitoring_network
    ports:
      - 9300: 9300

  kpi_connect_monitoring_alertmanager:
    image: prom/alertmanager:latest
    command: --config.file=/etc/alertmanager/alertmanager.yml
    ports:
      - 9093:9093
    volumes:
      - ./alertmanager/alertmanager.yml:/etc/alertmanager/alertmanager.yml

  kpi_connect_monitoring_node_exporter:
    image: prom/node-exporter:latest
    command:
      - '--path.rootfs=/host'
    volumes:
      - /:/host:ro,rslave
    networks:
      - kpi_connect_monitoring_network
    ports:
      - 9101:9101

networks:
  kpi_connect_monitoring_network:
    driver: bridge
    ipam:
     config:
       - subnet: 172.76.10.0/24
```

Рис. 3.6. Другий конфігураційний файл docker-compose.yml

```
version: '2'
services:

  kpi_connect_sonar_sonarqube:
    image: sonarqube:latest
    depends_on:
      - kpi_connect_sonar_postgresql
    environment:
      SONAR_JDBC_URL=jdbc:postgresql://kpi_connect_sonar_postgresql:5432/sonarqube
      SONAR_JDBC_USER: sonar
      SONAR_JDBC_PASSWORD: sonar
    volumes:
      - ../opt/sonarqube/conf:/opt/sonarqube/conf
      - ../opt/sonarqube/data:/opt/sonarqube/data
      - ../opt/sonarqube/extensions:/opt/sonarqube/extensions
      - ../opt/sonarqube/logs:/opt/sonarqube/logs
    networks:
      - kpi_connect_sonar_network
      - kpi_connect_overlay_network
    ports:
      - 9010:9010

  kpi_connect_sonar_postgresql:
    image: postgres:latest
    environment:
      - POSTGRES_USER=sonar
      - POSTGRES_PASSWORD=sonar
      - POSTGRES_DB=sonarqube
    volumes:
      - ../var/postgresql:/var/lib/postgresql
      - ../var/postgresql_data:/var/lib/postgresql/data
    networks:
      - kpi_connect_sonar_network
    ports:
      - 5432:5432

  kpi_connect_sonar_node_exporter:
    image: prom/node-exporter:latest
    command:
      - '--path.rootfs=/host'
    volumes:
      - /:/host:ro,rslave
    networks:
      - kpi_connect_sonar_network
      - kpi_connect_overlay_network
    ports:
      - 9102:9102

  kpi_connect_sonar_postgresql_exporter:
    image: wrouesnel/postgres_exporter:latest
    depends_on:
      - kpi_connect_sonar_postgresql
```

Рис. 3.7. Частина третього конфігураційного файлу docker-compose.yml



```
    environment:

      DATA_SOURCE_NAME="postgresql://sonar:sonar@kpi_connect_sonar_postgresql:5432/sonarqube?sslmode=dis
      able"
    networks:
      - kpi_connect_sonar_network
      - kpi_connect_overlay_network
    ports:
      - 9432: 9432

networks:
  kpi_connect_sonar_network:
    driver: bridge
    ipam:
     config:
       - subnet: 172.75.10.0/24
```

Рис. 3.8. Частина третього конфігураційного файлу docker-compose.yml

Таким чином, розроблені конфігураційні файли створюють основу роботи платформи, і відповідні контейнери можуть почати виконувати задачі сервісів, які на них запущені. Для правильної роботи кожного окремого компоненту

необхідно підготувати правильне налаштування його роботи та його складових. Розглянемо детальніше конфігурації основних засобів моніторингу і перевірки якості коду у наступних підрозділах.

3.2.2. Конфігурація моніторингового сервісу Prometheus

В якості основного конфігурації сервісу Prometheus виступає відповідний файл prometheus.yml. В ньому описується логіка основних компонентів, з яких сервіс збирає метрики із експортерів і встановлених клієнтських бібліотек для їх збереження у внутрішню базу даних.

Опис цього файлу показано на рис. 3.9-3.10. Складові полів цього файлу наступні.

Global – поле, у якому визначаються інтервали для збору даних метрик із системи. На даному прикладі визначені такі параметри, як *scrape_interval* – як часто збирати дані із досліджуваних елементів, *evaluation_interval* – як часто перевіряти правила, які визначають повідомлення і попередження про перехід деякого порогу. Параметр *external_labels* зазначає підписи, якими будуть підписуватися метрики при побудові повідомлень або графіків із відповідних сервісів збору.

Rule_files визначає розміщення файлу правил, за якими виконується побудова повідомлень і попереджень за допомогою системи Alert Manager.

Поле *scrape_configs* зазначає список окремих сервісів, з яких відбувається збір даних для моніторингу Prometheus. У параметрі *static_configs* зазначається адреса відповідного сервісу збору метрик, із відповідним відкритим портом. Крім того, *metrics_path* використовується для визначення шляху за протоколом HTTP, з якого Prometheus має збирати дані. За замовчуванням, це значення відповідає /metrics.

Поле alerting визначає окремі значення для підключення сервісу Alert Manager до заданої системи Prometheus. Конфігурація цього поля схожа із відповідними параметрами секції scrape_configs.

```yaml
global:
  scrape_interval:     15s
  evaluation_interval: 15s

  # Attach these labels to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
      monitor: 'kpi-connect-app'

# Load and evaluate rules in this file every 'evaluation_interval' seconds.
rule_files:
  - "alert.rules"

# A scrape configuration containing exactly one endpoint to scrape.
scrape_configs:
  - job_name: 'nodeexporter_app'
    scrape_interval: 5s
    static_configs:
      - targets: ['kpi_connect_app_node_exporter:9100']

  - job_name: 'nodeexporter_monitoring'
    scrape_interval: 5s
    static_configs:
      - targets: ['kpi_connect_monitoring_node_exporter:9101']

  - job_name: 'nodeexporter_sonar'
    scrape_interval: 5s
    static_configs:
      - targets: ['kpi_connect_app_node_exporter:9102']

  - job_name: 'cadvisor'
    scrape_interval: 5s
    static_configs:
      - targets: ['kpi_connect_app_cadvisor:9080']

  - job_name: 'prometheus'
    scrape_interval: 10s
    static_configs:
      - targets: ['kpi_connect_monitoring_prometheus:9090']

  - job_name: 'phpfpm_exporter'
    scrape_interval: 15s
    static_configs:
      - targets: ['kpi_connect_app_fpm_exporter:9153']

  - job_name: 'kpi_connect_symfony'
    scrape_interval: 15s
    metrics_path: '/metrics/prometheus'
    static_configs:
      - targets: ['kpi_connect_app_fpm:9001']

  - job_name: 'postgresql_app_exporter'
    scrape_interval: 15s
```

Рис. 3.9. Частина конфігураційного файлу prometheus.yml

Крім зазначеного файлу prometheus.yml, для правильної роботи відправки повідомлень і попереджень у сервіс Alert Manager, необхідно також визначити структуру файлу alert.rules. Він також складається із структури поле: значення, і в цілому вони не відрізняються між різними правилами.

Приклад оформлення правил для Alert Manager показано на рис. 3.11.

```
    static_configs:
      - targets: ['kpi_connect_app_postgresql_exporter:9432']

  - job_name: 'postgresql_sonar_exporter'
    scrape_interval: 15s
    static_configs:
      - targets: ['kpi_connect_sonar_postgresql_exporter:9432']

  - job_name: 'sonarqube'
    scrape_interval: 30s
    metrics_path: '/api/prometheus/metrics'
    static_configs:
      - targets: ['kpi_connect_sonar_sonarqube:9010']

  - job_name: 'minio'
    scrape_interval: 30s
    metrics_path: '/minio/v2/metrics/cluster'
    static_configs:
      - targets: ['kpi_connect_app_minio:9000']

alerting:
  alertmanagers:
  - scheme: http
    static_configs:
    - targets:
      - 'kpi_connect_monitoring_alertmanager:9093'
```

Рис. 3.10. Частина конфігураційного файлу prometheus.yml

```
groups:
- name: targets
  rules:
  - alert: monitor_service_down
    expr: up == 0
    for: 30s
    labels:
      severity: critical
    annotations:
      summary: "Monitor service non-operational"
      description: "Service {{ $labels.instance }} is down."

- name: host
  rules:
  - alert: high_cpu_load
    expr: node_loadl > 1.5
    for: 30s
    labels:
      severity: warning
    annotations:
      summary: "Server under high load"
      description: "Docker host is under high load, the avg load 1m is at {{ $value}}. Reported by instance
      {{ $labels.instance }} of job {{ $labels.job }}."

  - alert: high_memory_load
    expr: (sum(node_memory_MemTotal_bytes) - sum(node_memory_MemFree_bytes + node_memory_Buffers_bytes +
    node_memory_Cached_bytes) ) / sum(node_memory_MemTotal_bytes) * 100 > 85
    for: 30s
    labels:
      severity: warning
    annotations:
      summary: "Server memory is almost full"
      description: "Docker host memory usage is {{ humanize $value}}%. Reported by instance {{
      $labels.instance }} of job {{ $labels.job }}."

  - alert: high_storage_load
    expr: (node_filesystem_size_bytes{fstype="aufs"} - node_filesystem_free_bytes{fstype="aufs"}) /
    node_filesystem_size_bytes{fstype="aufs"}  * 100 > 85
    for: 30s
    labels:
      severity: warning
    annotations:
      summary: "Server storage is almost full"
      description: "Docker host storage usage is {{ humanize $value}}%. Reported by instance {{
      $labels.instance }} of job {{ $labels.job }}."
```

Рис. 3.11. Частина конфігураційного файлу alert.rules

Правила в рамках цього файлу зазначаються окремими групами. Для цього виділяється окреме поле *groups*, в рамках якого зазначається назва окремої групи name, і відповідний список правил під полем *rules*.

Назва повідомлення у списку правил зазначається під полем *alert*. Для визначення виразу, за яким виконується порівняння метрики щоб опублікувати повідомлення зазначається параметр *expr*. Перевірка за часом, під час якого виконання виразу є правдивим потрібно встановити у полі *for*. Поле *labels* визначає додаткові позначки, з яким встановлюється дане повідомлення. У даному випадку визначено поле серйозності проблеми *severity*. Параметр annotations зазначає підпис самого повідомлення, яке відповідає заданому правилу.

Для налаштування відправки повідомлень через Alert Manager, необхідно також налаштувати додатковий файл alertmanager.yml, який буде завантажуватися у контейнер із цим сервісом. Приклад реалізації такого конфігураційного файлу показано на рис 3.12:

```
global:
  resolve_timeout: 1m

route:
  receiver: 'email-notifications'

receivers:
- name: 'email-notifications'
  email_configs:
  - to: example@gmail.com, example2@gmail.com
    from: kpi-connect_monitoring@gmail.com
    smarthost: smtp.gmail.com:587
    auth_username: kpi-connect_monitoring@gmail.com
    auth_identity: kpi-connect_monitoring@gmail.com
    auth_password: password
    send_resolved: true
```

Рис. 3.12. Конфігураційний файл alertmanager.yml

Глобальне налаштування роботи сервісу відправки показане в рамках поля *global*. В рамках цього поля визначаються параметри за замовчуванням, необхідні для відправки повідомлень за вибраним каналом зв’язку. У даному

прикладі, *resolve_timeout* визначає час, необхідний Alert Manager для відправки повідомлення про кінець проблеми із пов’язаною деякою метрикою.

Параметри поля *route* визначає набір правил відправки, в рамках яких визначається окремі канали зв’язку. Для підключення окремого сервісу інтеграції повідомлень або прийому відповідних листів на електрону пошту у цьому файлі зазначається список таких значень під полем *receivers*.

В даному випадку, застосовування сервіс електронних повідомлень на пошту, який в рамках поточної версії Alert Manager має назву *email_configs*. Це поле передбачає ряд додаткових параметрів, необхідних для правильної відправки повідомлень. Параметри to і from показують адресу відправки, і адресу прийому куди слід відправляти різні повідомлення від Prometheus. Ряд інших полів (*smarthost, auth_username, auth_identity, auth_username, auth_password*) зазначає конфігурацію SMTP серверу, з якого буде виконуватися відправка листів. *Send_resolved* конфігурація передбачає відправку додаткового повідомлення, якщо якась подія вже завершилася.

Налагодження пакету Prometheus і Alert Manager обмежено відповідними файлами, показаними вище. Далі розглянемо конфігурацію сервісу Grafana, а також сервісу перевірки коду SonarQube.

3.2.3. Конфігурація сервісу візуалізації моніторингових метрик Grafana

Сервіс Grafana вже постачається налаштованим і готовим до роботи одразу після ініціалізації контейнеру. У відповідних полях docker-compose.yml можна було вказати пару значення логін-пароль для входу у інтерфейс Grafana, і скоригувати доступність для перегляду моніторингової інформації іншим користувачам платформи.

Тим не менш, за допомогою додаткових налаштувань можна одразу підготувати систему, з якої Grafana буде отримувати дані для візуалізації, а також готову панель із графіками метрик.

На рис. 3.13 показано конфігурацію файлу datasource.yml, який розміщується по шляху /etc/grafana/provisioning/datasources:

```yaml
apiVersion: 1

datasources:
  - name: Prometheus
    type: prometheus
    access: proxy
    orgId: 1
    url: http://kpi_connect_monitoring_prometheus:9090
    basicAuth: false
    isDefault: true
    editable: true
```

Рис. 3.13. Конфігураційний файл datasource.yml

У відповідності до його конфігурації маємо такі основні параметри, що зазначаються тут. Поле *apiVersion* визначає версію API самого файлу конфігурації. Список усіх джерел збору даних визначається після параметру *datasources*. Він передбачає поле назва джерела (*name*), тип або вид джерела збору (*type*), вид доступу до цього джерела (*access*), ідентифікатор заданого поля (*orgId*), посилання для доступу до джерела через HTTP (*url*), контроль за аутентифікацією із віддаленим джерелом даних (*basicAuth*), чи є це джерело інформації за замовчуванням (*isDefault*), і чи можливо редагувати відповідні настройки через графічний інтерфейс додатку (*editable*).

Для побудови основної панелі на основі заданого джерела інформації визначається окремий конфігураційний файл dashboard.yml, який розміщується по шляху /etc/grafana/provisioning/dashboard у тому ж контейнері Grafana. Приклад такого файлу показано на рис. 3.14:

```yaml
apiVersion: 1

providers:
  - name: 'Prometheus'
    orgId: 1
    folder: ''
    type: file
    disableDeletion: false
    editable: true
    allowUiUpdates: true
    options:
      path: /etc/grafana/provisioning/dashboards
```

Рис. 3.14. Конфігураційний файл dashboard.yml

Конфігурація цього файлу описується схожим чином, як і у файлі datasource.yml. Для визначення списку метрик, за яким завантажується панель із графіками використовується поле *providers*. Параметр *disableDeletion* визначає чи можливо видалити цю завантажену панель із файлу використовуючи графічний інтерфейс. Поля *editable* та *allowUiUpdates* застосовані для можливості видалення зазначеної панелі користуючись засобами графічного інтерфейсу. Поле *path* необхідне для завантаження окремих полей у деякій задані папці, визначеній в рамках цього конфігураційного файлу. В той же час, параметр *folder* необхідний щоб показати саму папку, де зберігається панель із графіками.

Файли панелей, які можуть завантажуватися у відповідності до конфігурації шляху із файлу datasource.yml мають набір метрик і відповідних даних, на основі яких будуються окремі графіки із даними. Ці файли мають тип даних JSON, і зазвичай є досить громіздкими через кількість можливих параметрів, які в них перераховані.

Типовий вигляд панелі моніторингу у Grafana має вигляд як на рис. 3.15. Дана панель показує стандартні метрики, які збираються у Prometheus за допомогою node-exporter, і вказують стан здоров’я хоста Docker, на якому запущений контейнер із експортером.

Рис. 3.15. Панель візуалізації даних Grafana для метрик зібраних із хоста Docker

3.2.4. Конфігурація сервісу перевірки якості коду SonarQube

Для налагодження роботи сервісу SonarQube в більшій мірі необхідно підготувати необхідну для її роботи окрему базу даних. У нашому прикладі для цього використовується база PostgreSQL, яка буде зберігати усі перевірки якості коду у відповідних таблицях Sonar. Налагодження основного конфігураційного файлу sonar.properties необхідне для підключення самої системи SonarQube до відповідної бази даних, і організації їх роботи.

Тим не менш, опис ключових полей для підключення до СУБД вже було виконано в рамках параметру environment у відповідному файлі docker-compose.yml, який було розглянуто на рис. 3.7. Серед розглянутих змінних

середовища, необхідними параметрами підключення є власне адреса бази даних (*sonar.jdbc.url*), користувач бази даних, який буде виконувати підключення до неї (*sonar.jdbc.user*) та пароль від цього користувача (*sonar.jdbc.password*).

Сам конфігураційний файл sonar.properties за замовчуванням має безліч окремих додаткових параметрів для більш контрольованого налагоджування роботи сервісу SonarQube. Для звичайної роботи в якості серверу прийому даних зі сканерів перевірки якості написання коду, стандартна конфігурація, яка йде у комплекті із образом для Docker, а також із визначеними змінними для підключення бази даних достатня для роботи серверу.

Крім серверу SonarQube необхідно розглянути сам сканер перевірки коду, який розміщується як окремий сервіс разом із веб-додатком розробки. При його розміщенні як додатковий пакет у тому ж контейнері із самим програмним кодом, необхідно створювати додатковий конфігураційний файл sonar-scanner.properties, в якому необхідно зазначати параметри для підключення до основного серверу (*sonar.host.url*), та зазначати токен аутентифікації, який створюється засобами SonarQube графічного інтерфейсу (*sonar.login*).

У розроблюваному проєкті так як сканер коду також розгортується за допомогою образу Docker у відповідний йому контейнер, додаткова конфігурація через окремий файл не потрібна. В такому разі достатньо визначити окремі змінні середовища контейнеру, в якому потрібно зазначити адресу серверу SonarQube, і токен аутентифікації. Приклад показано на рис. 3.5.

В результаті розглянутих конфігураційних файлів, необхідних для розгортання самого модуля навчально-методичного комплексу, та додаткових сервісів перевірки якості коду та моніторингу стану системи, маємо окрему структуру роботи платформи у загальному вигляді. На рис. 3.16 показано отриману структурну схему окремих модулів платформи, та їх зв’язок один із одним:

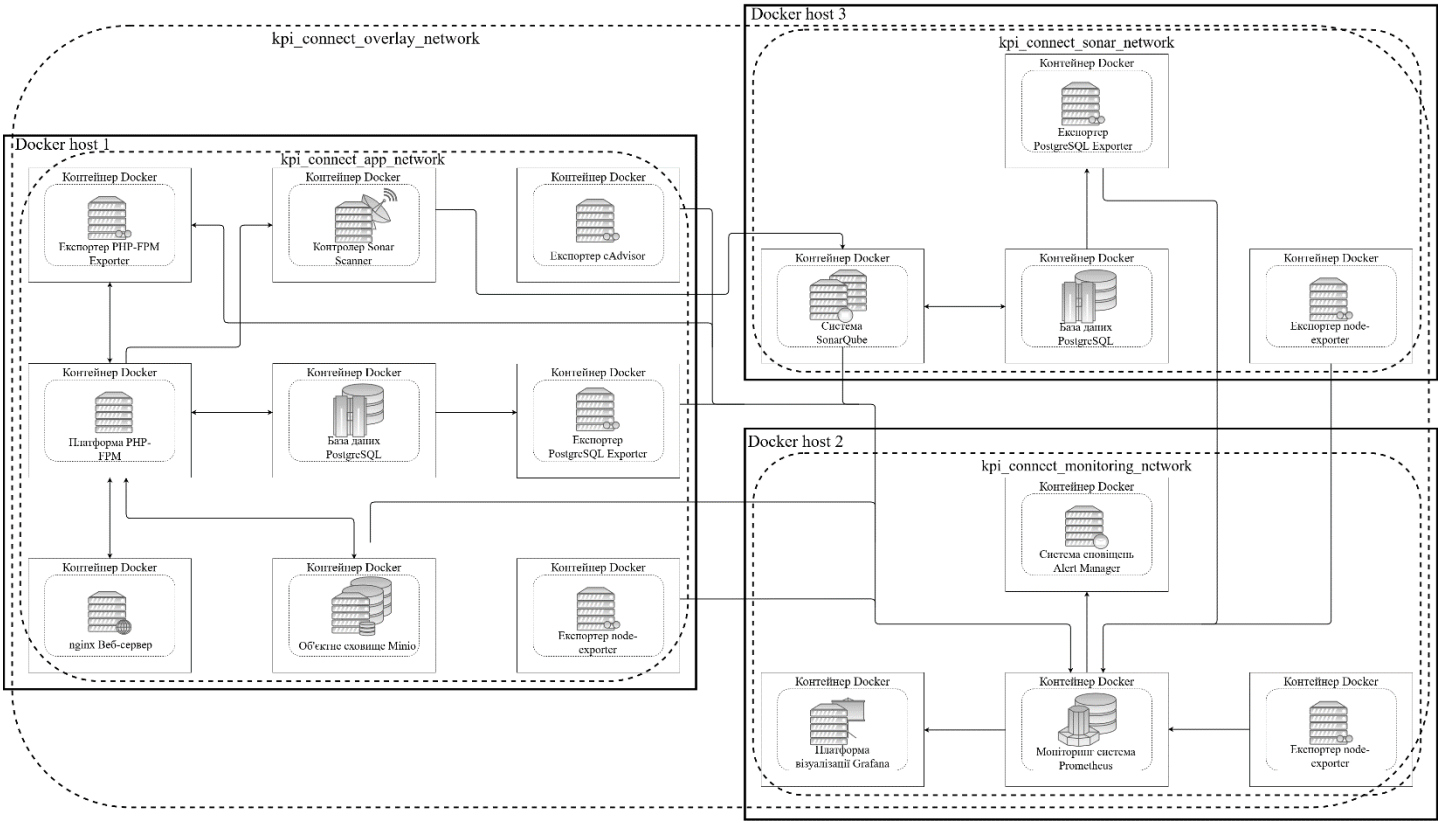


Рис. 3.16. Структурна схема складових навчальної платформи

Висновки до розділу 3

У відповідності до розглянутої моделі навчально-методичного комплексу було визначено окремі апаратні і програмні складові, необхідні для роботи платформи в цілому. Так, було обрано в якості основи роботи кожного окремого сервісу систему Docker із застосуванням технології розгортання декількох контейнерів Docker Compose.

Крім зазначених контейнерів Docker було обрано основні програмні технології, використанні для написання коду. Так, для роботи бізнес логіки на рівні backend було застосовано мову програмування PHP, при цьому було також обрано додатковий фреймворк із набором додаткових пакетів Symfony. Для роботи frontend частини платформи було вирішено застосовувати мову програмування JavaScript із відповідним фреймворком React, до якого також підключено ряд додаткових бібліотек для побудови інтерфейсу користувача.

Так як описані складові розроблені на програмному рівні, було детальніше розглянуто їх конфігурацію в рамках контейнерів Docker. Веб-сервер у платформі відповідає nginx, а основна база даних має назву PostgreSQL. Для правильної роботи бізнес логіки із nginx, програмна частина роботи виконується на окремому контейнері з FastCGI інтерфейсом, що має назву PHP-FPM.

Так як ціль роботи полягає також у оцінці роботи самого комплексу було визначено додаткові компоненти платформи, які надають можливість перевірки їх стану. На основі розглянутих технологій, на окремому Docker хості було визначено моніторинговий сервер Prometheus. Для візуалізації зібраних метрик використовується окремий сервіс, що має назву Grafana, а відправка повідомлень про стан системи реалізовано у контейнері Alert Manager. Перевірка якості написання програмного коду виконується за допомогою сервісу SonarQube, який також розміщено на окремому Docker хості.

В результаті було сформовано структурну схему окремих контейнерів платформи із різними Docker хостами. Результати тестування роботи моніторингових сервісів і системи перевірки якості коду буде досліджено у 4 Розділі даної роботи.

РОЗДІЛ 4
ТЕСТУВАННЯ І ОЦІНКА ЯКОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1. Тестування якості написання коду, і моніторинг складових платформи

На основі написаної структури моніторингу роботи системи, а також сервісу перевірки якості коду, можемо побачити як працює система із реальними метриками в процесі тестування її роботи.

4.1.1. Тестування основних метрик роботи платформи

Для отримання даних про стан роботи кожного окремого компоненту, як вже було зазначено раніше, скористаємося сервісом візуалізації даних Grafana, використовуючи дані з Prometheus. Приклад вже раніше побудованої одної з панель роботи Grafana було показано на рис. 3.15. Розглянемо детальніше приклади деяких метрик із системи під час її активної роботи.

На рис. 4.1 показано метрику навантаження роботи процесору в цілому на кожній окремій структурі платформи:

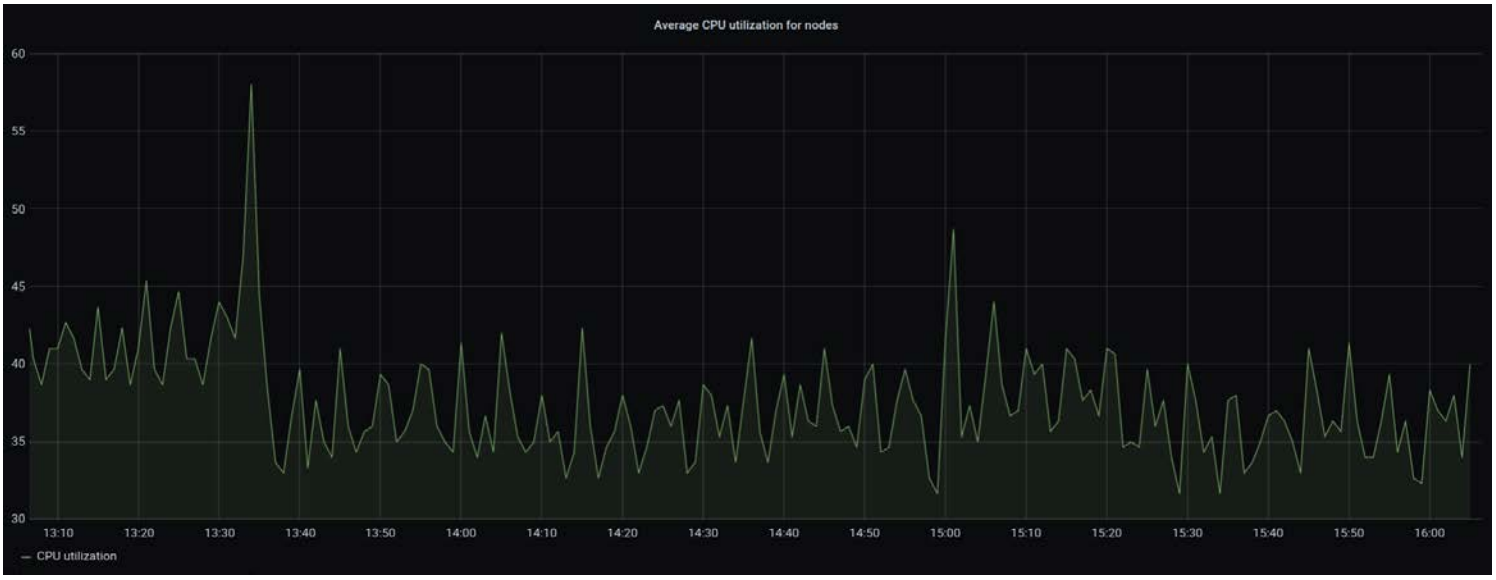


Рис. 4.1. Графік середнього навантаження роботи процесору на всіх компонентах платформи

Як можна побачити, значення графіку вказують, що центральний процесор з кожного окремого Docker хосту в середньому не є перевантаженим, адже

графіку вказують значення приблизно 45-40% від загальної можливості роботи процесору. Це дає змогу вказати на те, що існує запас у продуктивності роботи за обчислювальною потужністю із зростом оброблюваної інформації на окремих сервісах.

На рис. 4.2. показано кількість доступної оперативної пам’яті серед усіх застосованих Docker хостів:

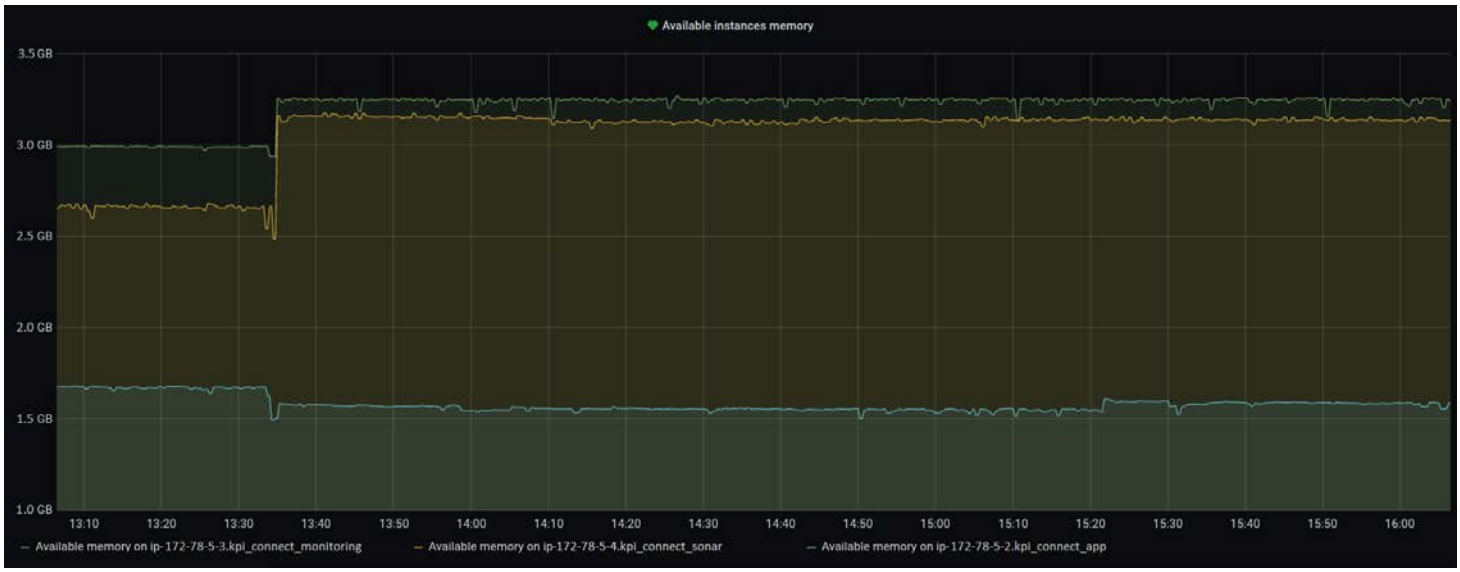


Рис. 4.2. Графік кількості доступної оперативної пам’яті на хостах

Виходячи із графіків, бачимо, що хост який виконує роботу програмного коду (kpi_connect_app) має меншу кількість вільної пам’яті, у порівнянні з іншими хостами. Тим не менше, значення у жодному графіку не досягає позначки 0 Гб, що означає всі хости мають також достатню кількість вільної додаткової пам’яті для виконання додаткових навантажень. При значені у 8 ГБ оперативної пам’яті в цілому, така кількість вільної пам’яті вважається достатньою, але можна її і апаратно збільшити.

На рис. 4.3 вказано графік кількості активних з’єднань із основою базою даних.

Графік складається із двох частин, і вказує роботу власне основної бази даних, яка використовується для у production версії платформи. Stage графік показано для показання роботи бази даних у тестовому середовищі. Приблизно рівна зміна кількості зв’язків із базою даних вказує на те, що обробка даних

виконується у звичайному режимі, і взаємодія компонентів і самої СУБД забезпечено без жодних помилок.

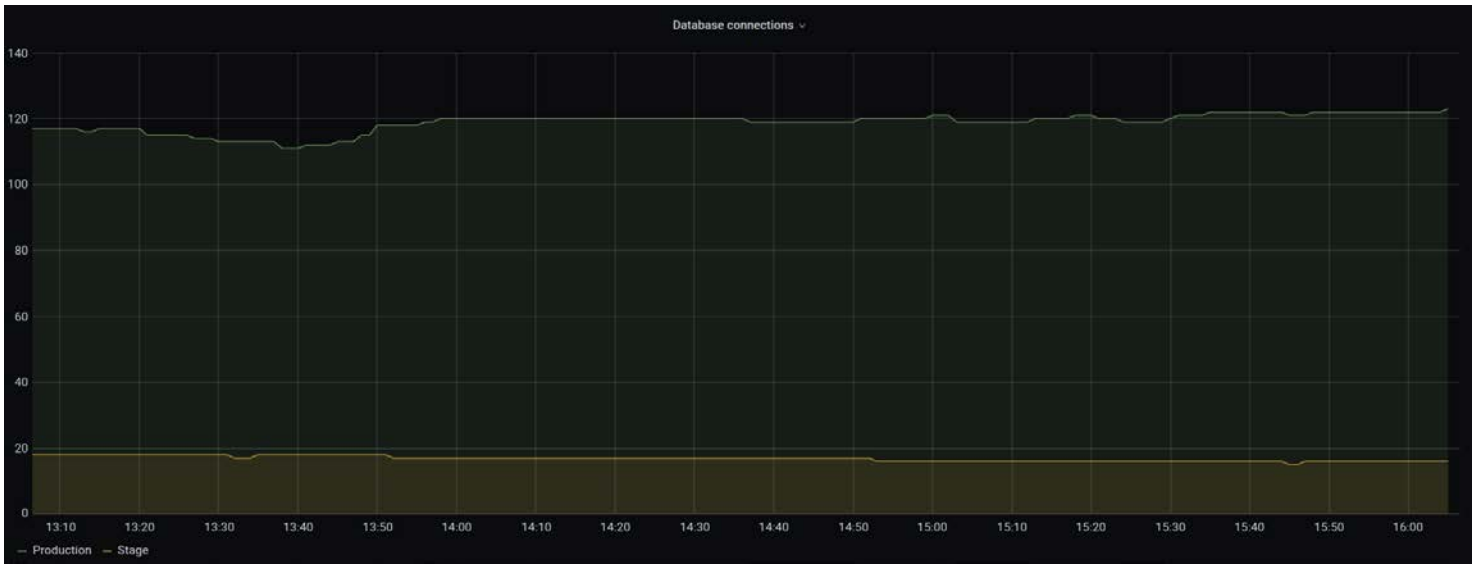


Рис. 4.3. Графік кількості активних з’єднань із основною базою даних

На рис. 4.4. показано кількість власне вільної оперативної пам’яті, яка наявна на контейнері бази даних:

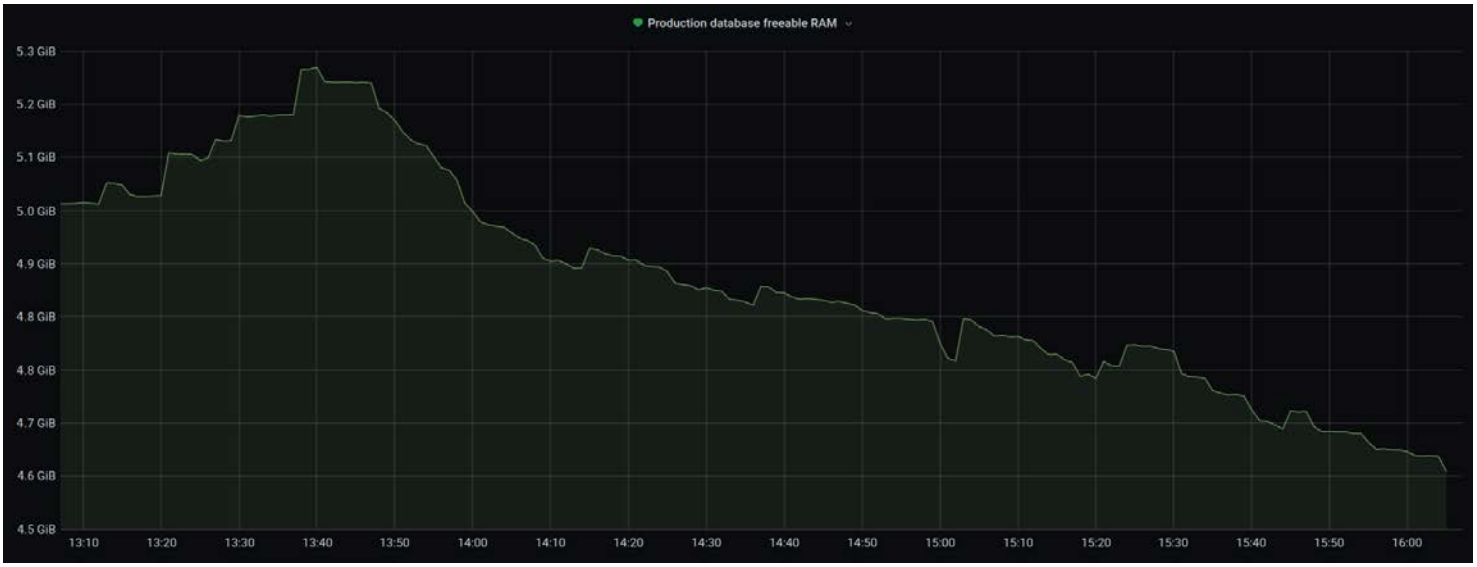


Рис. 4.4. Графік кількості доступної вільної пам’яті із бази даних

Відповідно до цього графіку можна побачити, що основна база даних використовує основну кількість оперативної пам’яті з хоста, на якому виконується програмний код платформи. Тим не менше, значення також приблизно зосереджено на 4-5 ГБ використовуваної оперативної пам’яті, при цьому значення може бути легко зменшено в залежності від активних з’єднань,

що не має викликати додаткових навантажень для роботи як і інших контейнерів, так і самого Docker хосту.

На рис. 4.5 вказано графік кількості використовуваного місця для зберігання даних в рамках контейнеру і хосту бази даних:

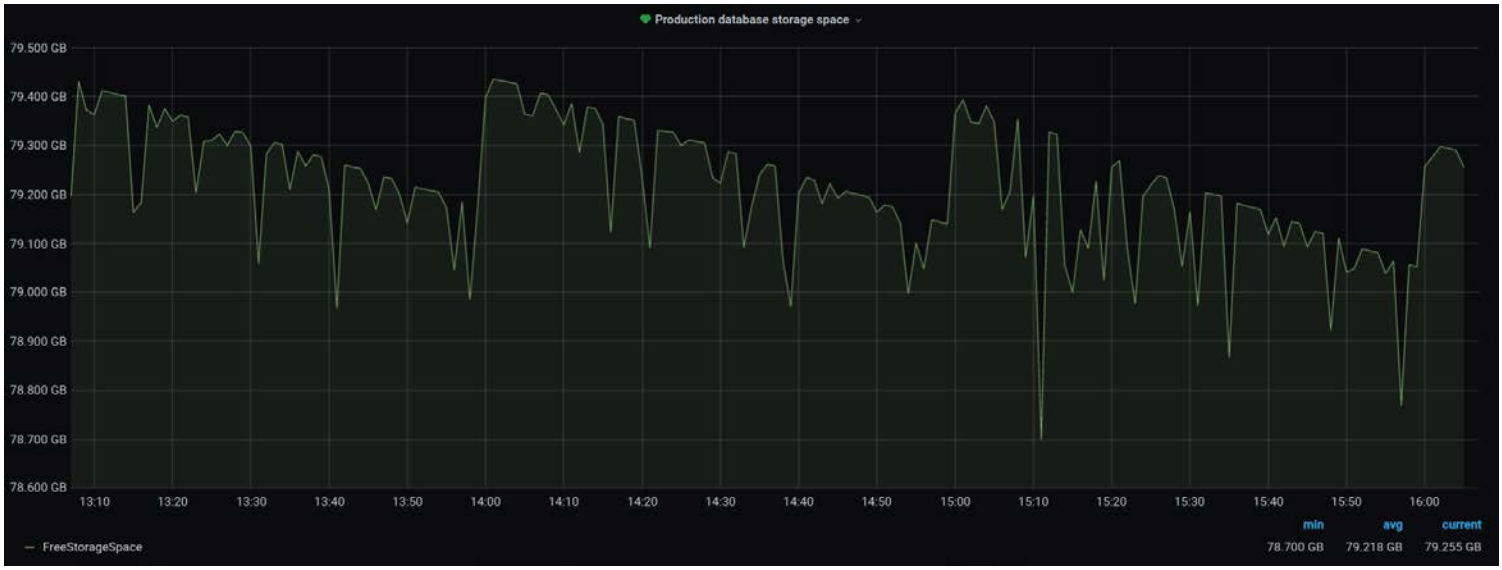


Рис. 4.5. Графік кількості вільного місця для зберігання бази даних

Даний графік вказує на метрику завантаженості віртуального жорсткого диску, виділеного під основну базу даних. Виходячи із його показників, можемо побачити, що графік кожну годину збільшується у середньому на 400 МБ, але при цьому постійно тримається в одних і тих же значеннях. Збільшення графіку досягається за рахунок автоматичного скрипту, який встановлено у контейнері PostgreSQL, що перезавантажує сервіс бази даних, і використовує команду */proc/sys/vm/drop_caches* для очистки об’єктів, які більше не використовується. За рахунок цієї операції досягається більша стабільність роботи бази даних, і очистка місця на жорсткому диску від старої обробленої інформації.

На рис. 4.6. можна побачити кількість HTTP трафіку, який проходить через систему.

Дана метрика також розподіляє сам трафік за відповідними HTTP кодами, які вказують чи був успішний запит і відповідь до платформи системи, або виникла деяка помилка на стороні клієнту, або серверу. Як можна побачити із графіку, він має особливість генерації трафіку у вигляді «зубців», які періодично збільшуються через кількість запитів до системи, але в цілому розміщуються на

позначці у приблизно 100 запитів за хвилину. Така поведінка пов’язана із фактом того, що система для користувачів не потребує оновлення кожної хвилини, адже необхідний деякий час для перегляд інформації на окремій сторінці платформи.

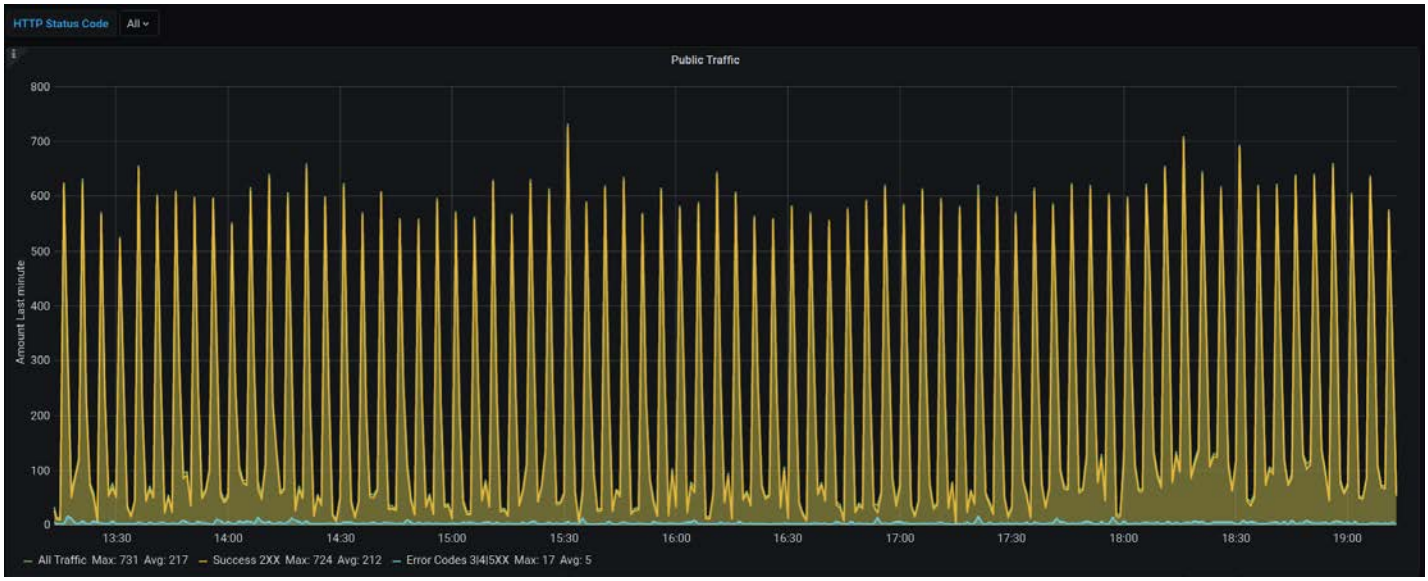


Рис. 4.6. Графік кількості публічного HTTP трафіку, який проходить через систему

Графік, який можна побачити на рис. 4.7. вказує на кількість активних з’єднань із системою на поточний момент часу:

Рис. 4.7. Графік кількості активних з’єднань із платформою у поточний момент часу

У відповідності до цього графіку можна сформувати кількість активних користувачів системи, які мали доступ до платформи у певний проміжок часу під час тестування. Як видно із характеру зміни графіка, у вечірній час кількість

з’єднань збільшується, що підтверджує факт того, що систему використовують в більшості в той час, коли користувачам зручніше працювати за комп’ютером, смартфоном тощо для перегляду відповідної інформації платформи.

Як вже було показано на рис. 4.6, даний графік вказував на загальну кількість трафіку, який проходить через систему, із розподілом на різні HTTP коди відповіді обробки запитів. На рис. 4.8 виокремлено окрему діаграму, яка вказує на кількість помилкових HTTP за декілька хвилин до поточного часу:

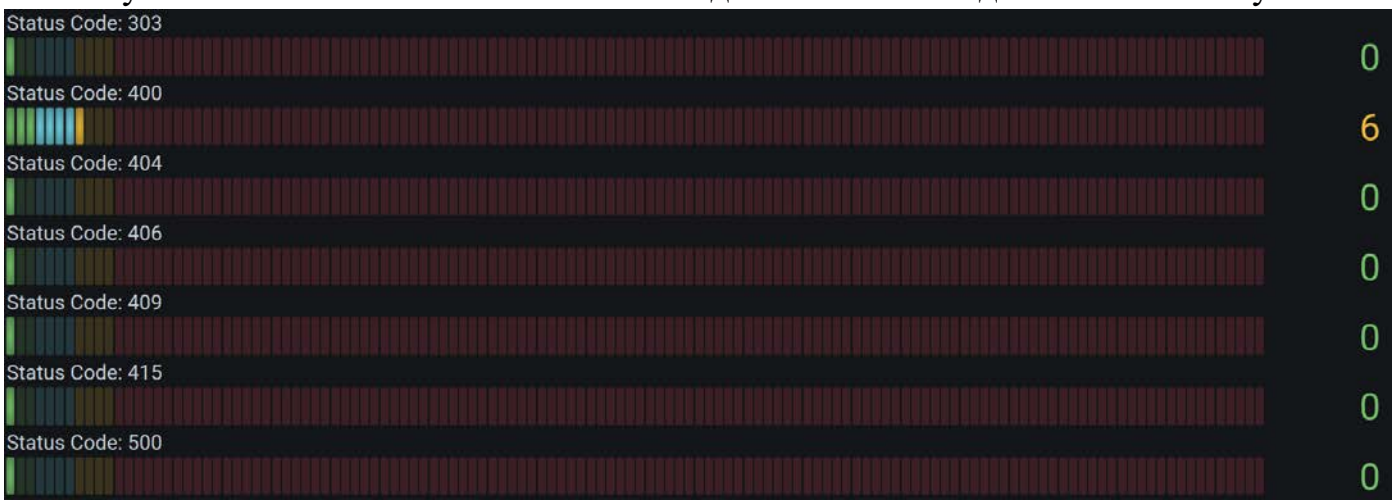


Рис. 4.8. Діаграма кількості помилкових HTTP запитів із відповідним статус кодом

У даній діаграмі показані типові помилки, які можуть надходити в якості статус коду з HTTP трафіку. На зображеному рисунку можна побачити, що за останні декілька хвилин платформа отримала декілька запитів із кодом 400 Bad Request – Помилковий запит. Це вказує на те, що клієнт не зміг передати запит на сервер через незрозумілий синтаксис, тому з’єднання не було встановлено. Кількість інших статус кодів вказує на те, що мережа платформи є у робочому стані, і приймає трафік стабільно.

Цей ряд метрик було отримано за допомогою відповідних експортерів для бази даних PostgreSQL Exporter, експортеру і пакету передачі даних на Prometheus – Symfony Prometheus Metrics Bundle, а також PHP-FPM Exporter. Крім того, у графіках наявні окремі метрики роботи стану контейнерів, отриманих засобами node-exporter і cAdvisor.

В подальшому розглянемо результати тестування якості написання коду, застосовуючи сервіс SonarQube.

4.1.2. Тестування якості написання коду платформи

Для перевірки стану програмного коду, скористаємося інструментом SonarQube, який розміщено на окремому хості Docker. Для аналізу самого тексту коду було застосовано окремий сканер Sonar Scanner, розміщений як окремий сервіс на хості Docker, де розташована власне програмна частина платформи.

На рис. 4.9. показано інтерфейс основного вікна роботи в SonarQube:

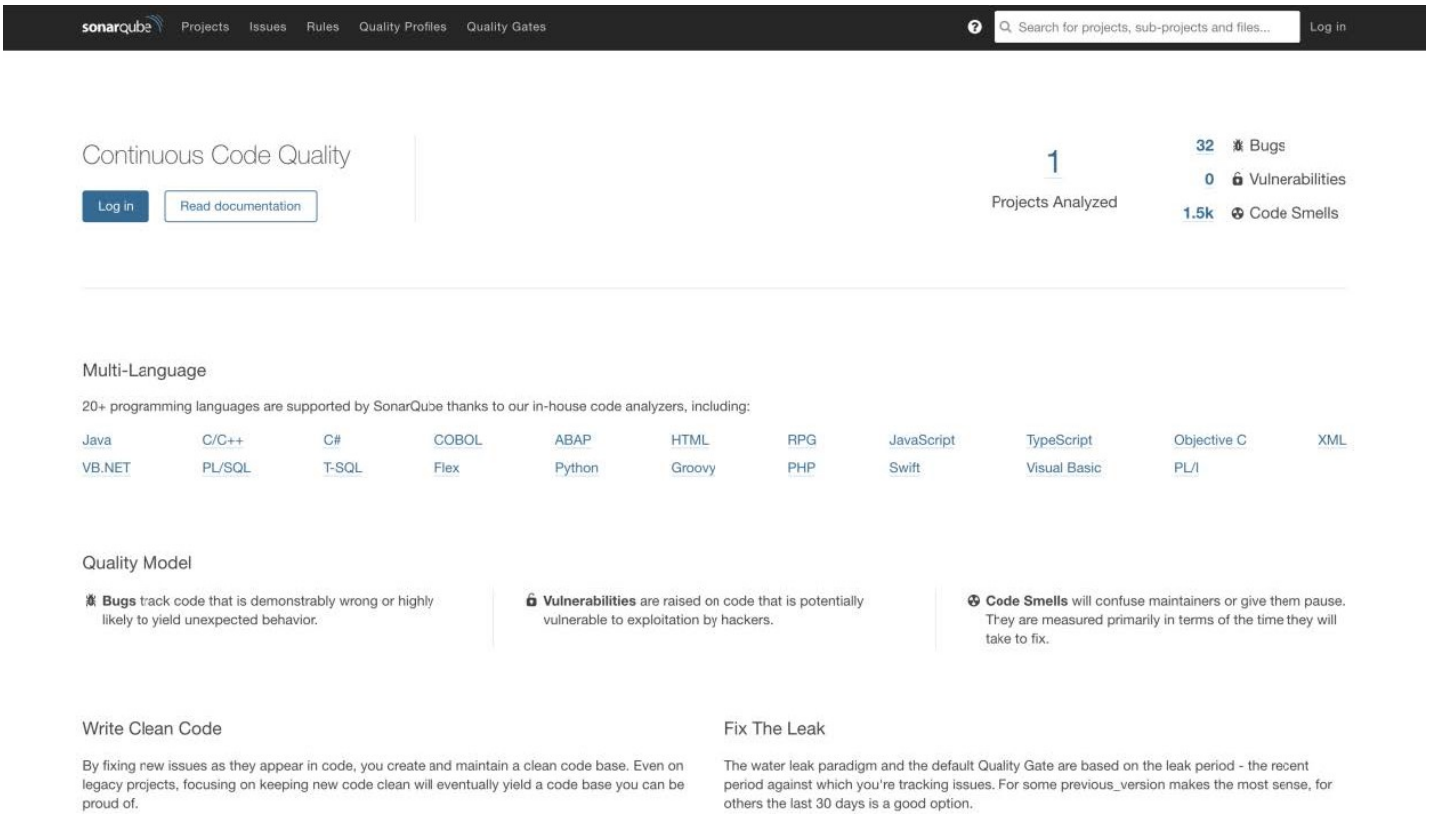


Рис. 4.9. Основний інтерфейс роботи із якістю коду SonarQube

Як можна побачити вже з цього вікна, SonarQube вже містить один проєкт програмного коду ,в якому було визначено 32 баги, 0 вразливостей і 1500 не зрозуміло написаного коду. Також нижче вказано пояснення щодо вказаних знайдених помилок або попереджень у коді. Розглянемо складові проєкту детальніше далі.

На рис. 4.10. показано сам проєкт роботи із відповідними значеннями:

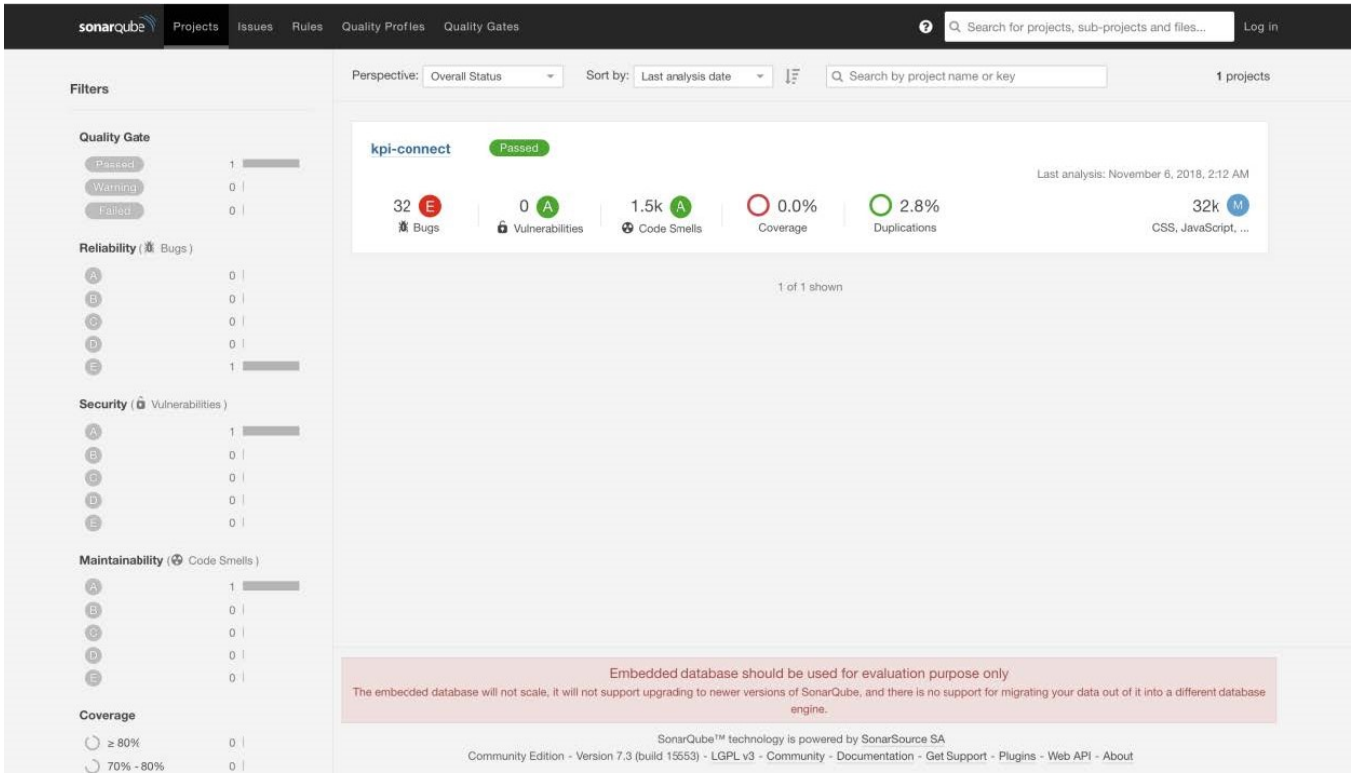


Рис. 4.10. Інтерфейс системи із інформацією про поточний стан проєкту

На цьому екрані вже детальніше вказано стан якості написання коду, його вразливості, а також стиль оформлення роботи.

Рис. 4.11 показує на список багів, які наявні у поточній версії програмного коду:

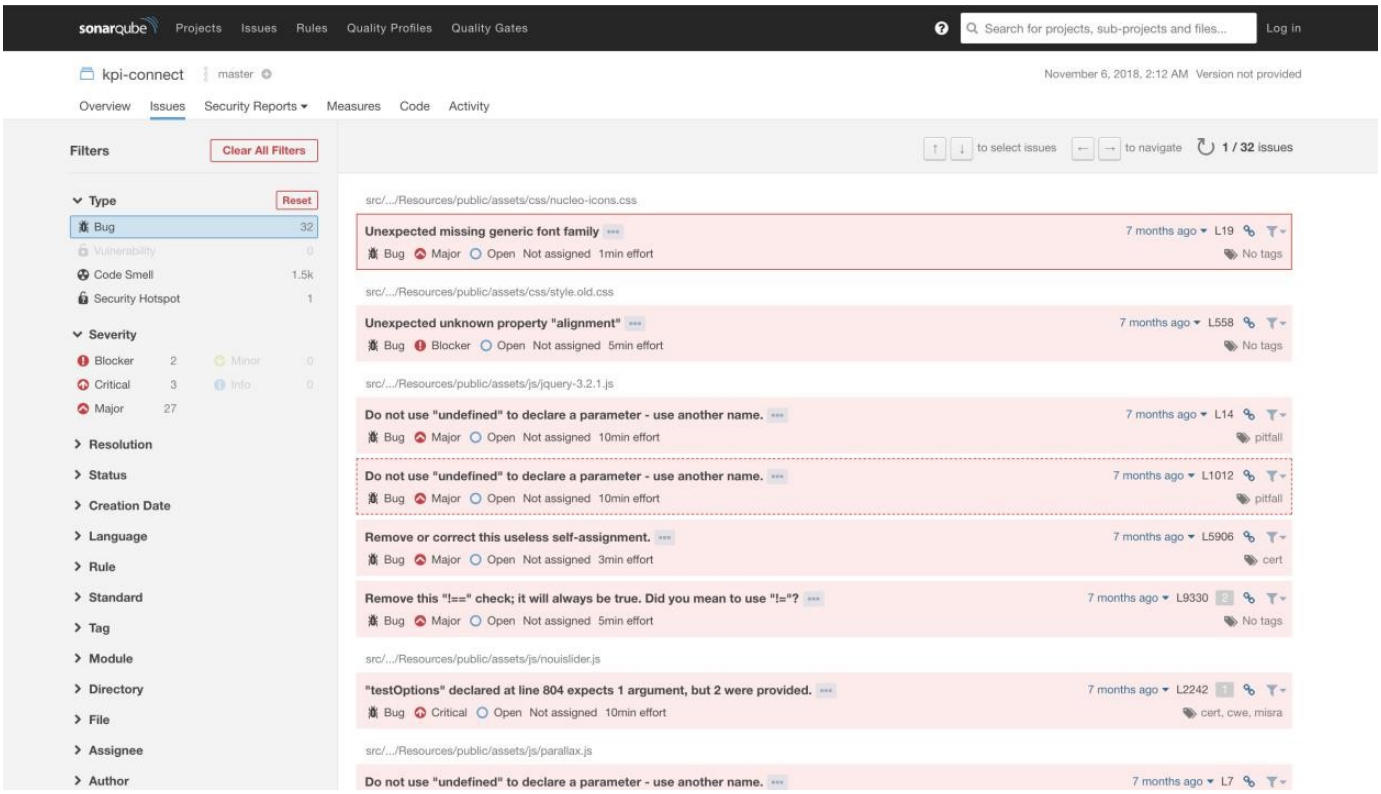


Рис. 4.11. Список багів у поточній версії програмного коду

Як можна побачити із відповідних значень багів, сканер коду перевіряє деякі фрагменти з коду, і вказує на потенційні помилки, які можуть з них йти. Слід врахувати, що сканер також виконує перевірку помилок у експортованих бібліотек програмування, необхідних для написання коду програми. В такому разі, слід ретельно переглядати список перевірки стану коду, і зіставляти із робочими файлами і класами проєкту платформи.

На рис. 4.12 зображено окремий показник, що має назву Technical Dept – технічний борг.

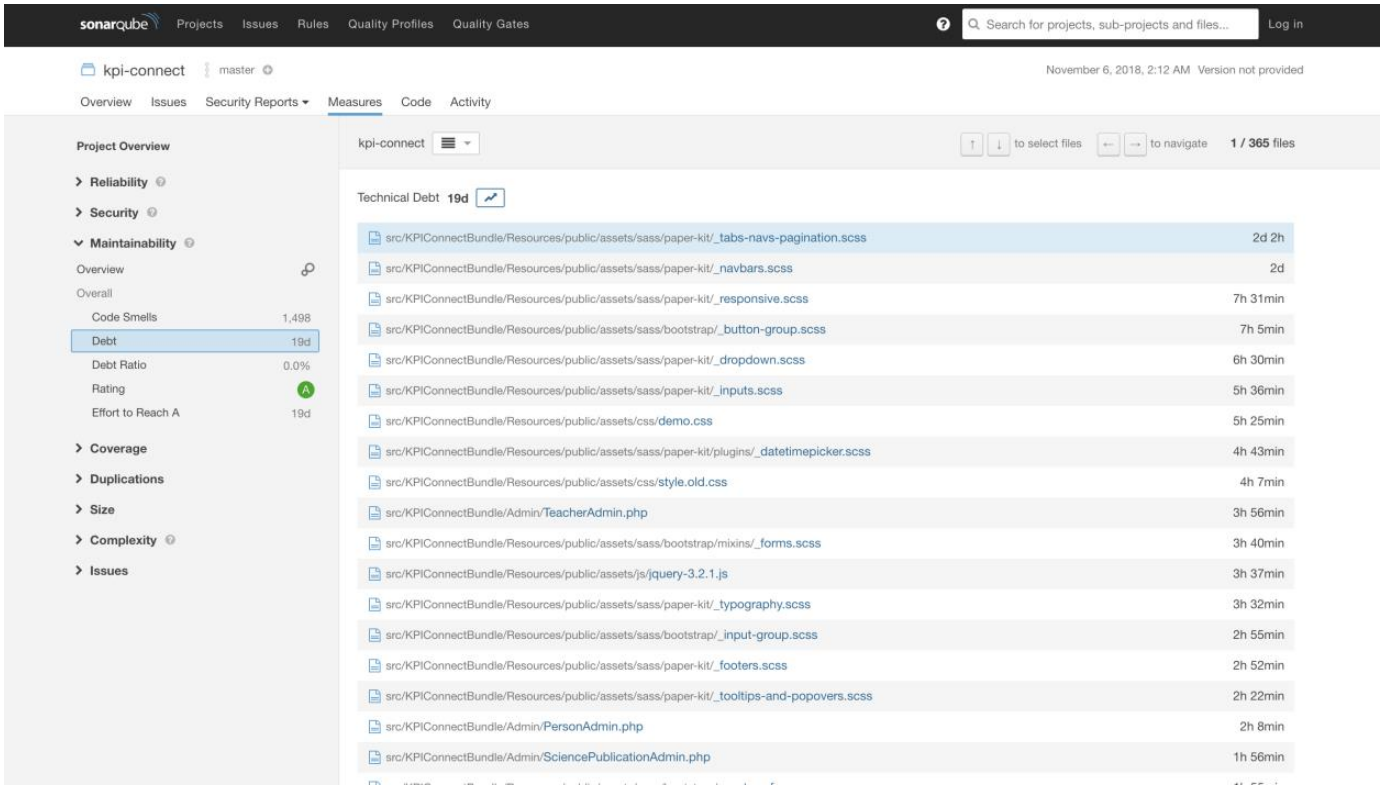


Рис. 4.12. Стан технічного боргу платформи розробки

Показник технічного боргу обраховує кількість неточностей програмного коду, який обраховується в окремій графі Code Smells. Він визначає час, необхідний для того, щоб виправити усі неточності у коді в різних файлах проєкту. Слід також зазначити, що Sonar Scanner виконує перевірку усіх додаткових бібліотек і пакетів, які йдуть у комплекті програмного коду, і в такому разі зазначає помилки і стан технічного боргу у відповідності до цих окремих елементів.

Рис. 4.13 містить діаграму огляду стану дубльованих фрагментів коду проєкту:

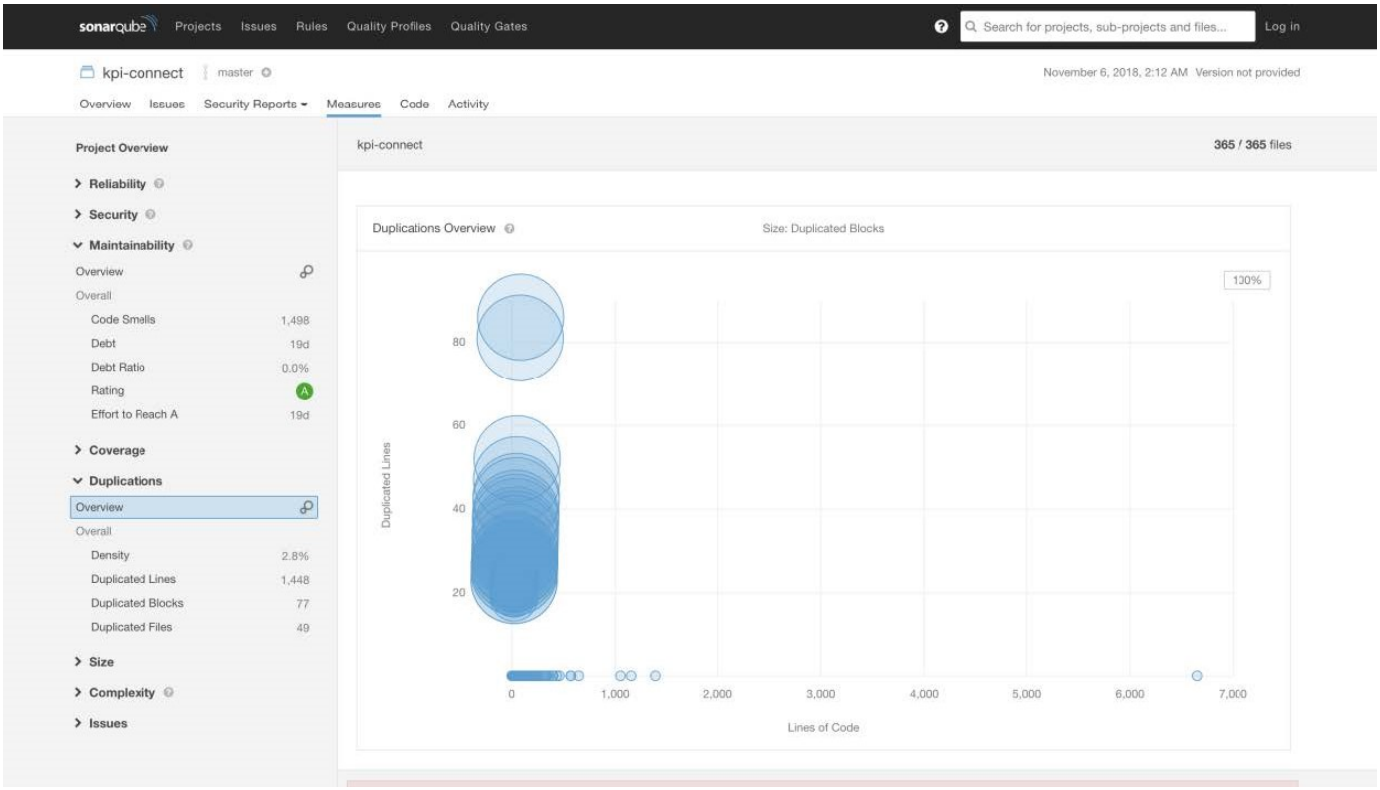


Рис. 4.13 Діаграма огляду дубльованих частин коду проєкту

Дана діаграма схематично показує розмір дубльованих частин коду у відповідності до кількості ліній коду до самої частини досліджуваного фрагменту. На основі цієї діаграми можна зробити висновок у наявності схожих елементів у програмному коді, які можна в подальшому замінити через окремий клас або бібліотеку функцій, які будуть визначати схожий інтерфейс роботи дубльованого коду. Але, слід зазначити що дубльований код у цьому аналізі може також містити автоматично згенеровані файли, або схожу структуру написання того, чи іншого функціоналу, тому слід детальніше розглядати окремі інші складові цього розділу SonarQube.

Як приклад, на рис. 4.14 показано аналіз щільності дубльованого коду серед робочого в цілому.

Щільність дубльованого коду розраховується через просте рівняння – кількість ліній дубльованого коду / кількість ліній всього коду * 100.

У відповідності до отриманого результату, можемо зробити висновок, що дубльованість коду у програмному коді має досить мале значення, у порівнянні до всього проєкту. Крім того, якщо переглянути список файлів, які містять дубльований код, можна побачити що всі вони відповідають файлам міграції

стану бази даних за допомогою системи ORM Doctrine. Заданий файл визначає узагальнену структуру для відправки моделей зміни таблиць і полів самої бази даних, тому наявність однакового коду між цими файлами визначена специфікацією самої бібліотеки програмування.

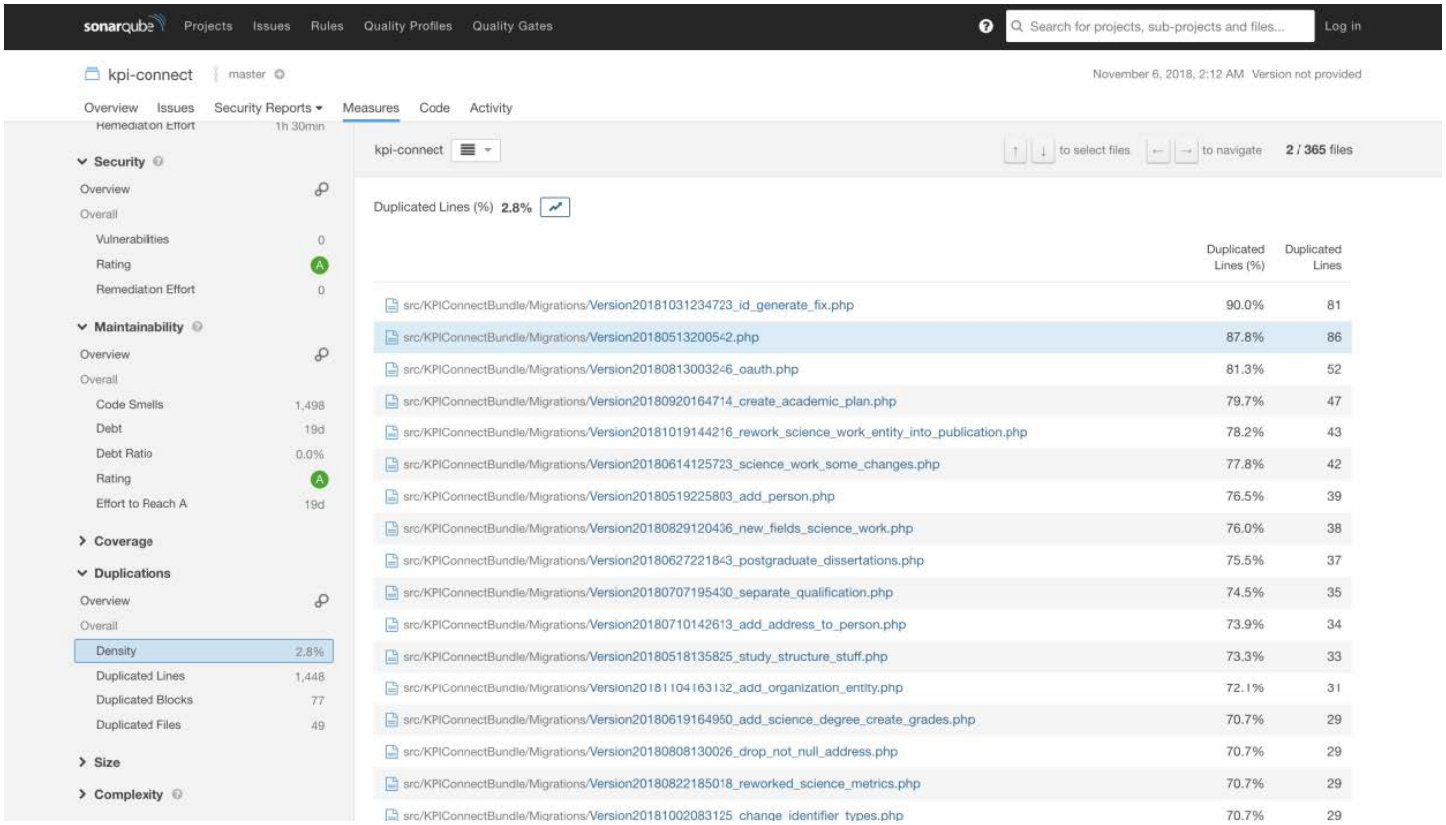


Рис. 4.14. Огляд файлів із дублюванням коду, та його щільність у порівнянні із усім кодом проєкту

4.2. Оцінка якості роботи розробленої платформи

Для оцінки якості роботи деякого програмного забезпечення скористаємося деякими існуючими характеристиками. У відповідності до стандарту ISO 25010:2011 [42], оцінка якості програмного продукту може виконуватися за 8 наступними показниками:

- Функціональність. Показник функціональності вказує на те, чи виконує програма поставлені на неї задачі, відповідає потребам її користувачів. Також, цей показник враховує аспект роботи всіх компонентів, їх сумісність один з одним;

- Надійність. Визначає характеристику безперебійної роботи програмного продукту за деякий визначений час;
- Зручність користування. Цей параметр характеризує ступінь зручності користування продуктом для її користувачів, його легкість засвоювання і експлуатації;
- Ефективність. Дана характеристика визначає ступінь продуктивності роботи програми при якомусь заданому наборі значень;
- Зручність супроводу. Цей параметр визначає простоту аналізу, тестування, моніторингу компонентів системи, а також можливість їх адаптувати до нових умов;
- Портативність. Ця характеристика показує ступінь легкості переносу продукту на іншу платформу, а також можливість роботи програми на різних робочих засобах – комп'ютерах, ноутбуках, смартфонах, планшетах, тощо;
- Сумісність. Цей показник вказує на ступінь інтеграції і роботи різних компонентів програми між собою. Дуже часто розподіляється на дві під характеристики: співіснування компонентів у продукті, і власне сумісність компонентів розподіляти інформацію між різними частинами програми;
- Захищеність. Цей параметр визначає степінь мінімізації загроз, які можуть бути пов'язані з несанкціонованим доступом до перегляду інформації, виходом зі строю технічних засобів програми, або некоректним використанням користувачами самого програмного продукту [4, 27].

Виходячи із розробленої структури платформи, можемо дати оцінку роботи платформи за кожною окремою розглянутою характеристикою.

Функціональність.

Серед поставлених задач у розробці навчально-методичного комплексу платформи автоматизації навчального процесу, можна назвати наступні розроблені компоненти, які формують структуру її роботи:

- Модель організації навчання у відповідності до положення про навчальний процес:
- Можливість внесення студентів у відповідні навчальні групи, та перегляд успішності навчання за визначеними дисциплінами;
- Застосування дисциплін навчання у відповідності до навчальних планів університету;
- Збереження навчально-методичного матеріалу у платформі, із можливістю надавати його здобувачам освіти для проходження навчання;
- Інтеграція інформації про користувачів із популярними соціальними сервісами.

Існує також ряд елементів платформи, який потребує подальшої розробки щоб повністю відповідати потребам користувачів у роботі із нею. Серед такого функціоналу можна включити необхідність доопрацювання графічного інтерфейсу користувачів платформи, із винесенням додаткових зовнішніх функцій.

Надійність.

Задану характеристику оцінки якості можна сформувати на основі отриманих результатів дослідження роботи платформи. Як можна було побачити із різних графіків і метрик, система є досить стійкою до падінь того, чи іншого її компоненту. Крім того, із відповідних графіків можна побачити, що у різних сервісів платформи існує запас у продуктивності роботи апаратних складових системи, що у свою чергу не призведе до помилок у роботі за відсутності зайвої оперативної пам’яті, чи ресурсів процесору.

В той же час, слід зазначити для цього питання, що існують ряд і недоліків, які відповідають даному параметру. Як можна було побачити із результатів роботи SonarQube, на даний моменту проєкті є ряд багів, які можуть впливати на процес роботи програми. Але, зміни при написані програмного коду в подальшому можуть як і вирішувати наявні проблеми в коді, так і створювати нові.

Зручність користування.

Поточний інтерфейс системи має доволі зручне розміщення основного його функціоналу, і надає користувачам прості засоби користування продуктом. Зміна даних у відповідності до розробленого інтерфейсу відбувається без суттєвих затримок, а перегляд отриманих результатів обробки даних, пов’язаних із навчальним процесом, надає відповідні результати у тому ж вигляді, як вони внесені у платформу.

Використання сучасних бібліотек побудови інтерфейсу Material UI допомогло додати ефективного планування робочих функцій користувачам для доступу у систему. Це також у свою чергу надало можливість виглядати увесь графічний інтерфейс у відповідності до сучасних норм дизайну веб-додатків.

Ефективність.

Ефективність роботи розробленої платформи можна визначити у відповідності до досліджуваних метрик роботи, отриманих із клієнтів сервісу Prometheus. Як можна побачити на прикладі рисунків завантаженості роботи бази даних, а також кількості вхідного трафіку на обробку даних, можна зробити висновок, що за деякий проміжок часу, система не показала жодних великих змін у плавності зміни графіків. Це вказує на те, що система справляється із поставленою на неї задачею досить ефективно. Тим не менше, перевірка великою кількістю запитів на обробку даних не було зроблено, і зробити остаточний аналіз ефективності роботи платформи із великим і максимальним навантаженням виявилося неможливим на поточний момент. Отримані результати вказують продуктивну роботу розробленого комплексу за звичайних або робочих умов, і відповідають показнику середнього навантаження.

Зручність супроводу.

Використання елементів моніторингу складних інформаційних систем Prometheus, візуалізація даних на основі оброблених метрик дослідження Grafana, а також застосування окремої системи перевірки написання коду SonarQube надали можливість платформі легко контролювати стан роботи компонентів, виконувати тестування стану написання коду, а також вносити

корективи у роботу цих складових. Крім того, використання великої кількості окремих клієнтів дослідження стану усіх компонентів (експортери, бібліотеки програмного коду інтеграції з Prometheus) надає можливість гранульованого контролю за продуктивністю роботи платформи в цілому, при цьому візуалізація даних за допомогою графіків у реальному часі дозволяє наочно отримати відповідну інформацію про ці складові.

Портативність.

У відповідності до цієї характеристики, можна визначити можливість перенесення усієї розробленої платформи на іншу інфраструктуру. Так як в основі роботи застосований принцип контейнеризації Docker, існує доволі проста можливість перенести всі складові роботи системи на іншу апаратуру. Застосування лише набору конфігураційних файлів, а також власне написаного програмного коду є достатньою необхідністю для переносу і розгортання усієї платформи. Нові контейнери, створені на інших хостах Docker, матимуть такі ж можливості з обробки даних роботи самої бізнес логіки, а також будуть застосовані ті ж операційні дані, які було попередньо перенесено в рамках міграції проєкту.

Як інший аспект цієї характеристики, слід зазначити, що застосування фреймворку для написання frontend частини проєкту React, дозволяє побудувати інтерфейс, який може працювати ефективно на різних системах і розмірах екрану відповідно. Було встановлено, що система працює правильно як і на екранах смартфонів, так і звичайних стаціонарних комп’ютерів, і ноутбуків.

Сумісність.

Дана розроблена система вже початкового постає із концепцією розробки різних компонентів, які можуть гармонійно працювати один із одним. В цьому, для розробки проєкту також допомогла мікросервісна архітектура додатку, в якому наявна велика кількість окремих функціональних частин, які виконують свої визначені елементи роботи платформи. Кожний окремий сервіс платформи взаємодіє з тою, чи іншою складовою системи у відповідності до моделі, зображеної на рис. 2.5.

Крім цього, інтеграція кожного окремого компоненту один із одним працює без жодних проблем, так як окремі сервіси платформи в цілому залежать від роботи інших його елементів. Наприклад, робота об’єктного сховища у системі залежить від бізнес логіки, яка передбачає доступ до сховища і процеси збереження і отримання навчальних файлів із неї.

Слід зазначити, що також існують ряд елементів, які потребують доопрацювання логіки інтеграції компонентів один із одним. Але, в цілому сумісність системи є досягнутою ціллю проєкту.

Захищеність.

Захищеність розробленої платформи досягається за допомогою застосування протоколів шифрування при обробці даних як із користувачами, так і з різними компонентами. Для цього, було застосовано принцип шифрування трафіку у HTTPS, користуючись сертифікатом від сервісу сертифікації Let’s Encrypt, який надає можливість звичайним користувачам налаштувати шифрування у своїх веб додатках.

Щодо забезпечення безпеки інших компонентів, під час тестування не було досліджено жодних критичних виходів зі строю самих елементів платформи, або її складових. Мережа доступу для роботи із окремим внутрішніми компонентами немає публічного доступу до Інтернет, і може бути доступною виключно через самі хости Docker Це визначає, що система є на поточний момент досить захищеною, і такою, що може протистояти на деякому рівні від вразливостей через неправильну конфігурацію середовища.

У відповідності розглянутих характеристик перевірки якості роботи програмного продукту, виконаємо порівняльну характеристику із рядом схожих продуктів. На даному прикладі, застосуємо порівняння із рядом інших навчально-методичних платформ, що надають схожий функціонал роботи. Для цього, в якості порівняння будуть показані популярна система Moodle із відкритим кодом, програма для проведення комп’ютерних тестів UniTest System, онлайн платформа для проведення тестів «Конструктор тестов», Система

тестування «INDIGO», а також пакет програм для проведення навчання і перевірки знань «VeralTest» [10].

Отримані результати дослідження показано у таблиці 4.1 із відповідними полями значень оцінки якості роботи систем. Оцінку роботи кожної із платформ було зроблено за 5 бальною шкалою, де 1 відповідає незадовільному результату за вибраною характеристикою, а 5 – дуже гарний результат.

Таблиця 4.1. Порівняльна характеристика оцінки якості роботи систем автоматизації навчального процесу

№	Назва характеристики	KPI-Connect	Moodle	UniTest System	«Конструктор тестов»	INDIGO	VeralTest
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	Функціональність	3	4	3	2	4	3
2	Надійність	4	5	3	2	3	3
3	Зручність користування	5	4	4	4	3	3
4	Ефективність	4	4	4	5	4	4
5	Зручність супроводу	5	3	4	3	2	5
6	Портативність	5	4	4	4	5	5
7	Сумісність	5	3	4	3	5	4
8	Захищеність	4	4	3	3	4	5

Як можна побачити із результатів порівняльної таблиці, розроблена система має в середньому кращі показники серед аналогічних до неї програм тестування, хоча ряд характеристик на деяких позиціях і програє конкурентам. Тим не менш, даний результат показує хороший результат аналізу розробки даного програмного забезпечення.

Висновки до розділу 4

На основі розробленої схеми роботи окремих компонентів платформи навчально-методичного комплексу було проведено тестування її роботи, і записані результати дослідження метрик системи. Крім того, було також проведено аналіз роботи сервісу перевірки якості коду, і завантажено поточний стан програмного коду платформи для отримання результатів тестування із відповідного сервісу SonarQube.

Отримані метрики системи Prometheus, які були візуалізовано за допомогою засобів сервісу Grafana, показали стабільність роботи платформи. В рамках дослідження було перевірено завантаженість Docker хостів на предмет оперативної пам’яті і проценту роботи центральних процесорів. Результати, отримані з відповідних метрик вузлів системи, вказують про наявність доступної вільної оперативної пам’яті, та можливого розширення обчислювальної можливості контейнерів, що дозволяє платформі працювати без проблем навіть при більшій кількості активних користувачів, і запитів на обробку даних. Крім того, показники роботи основної бази даних, а також контейнеру із бізнес логікою продукту вказують на нормальний стан роботи платформи.

В результаті перевірки якості написання коду веб-додатку, отриманий аналіз у системі SonarQube показав хороший стан введення коду. Навіть за наявності ряду багів, які у більшій мірі сканер отримав із додаткових бібліотек, а також деякої кількості неточностей коду, отримані результати показують ефективну розробку і процес написання проєкту.

Провівши ряд цих досліджень, було також зроблено аналіз оцінки якості роботи платформи, у відповідності до його специфікації. Для наглядної перевірки результатів аналізу було побудовано порівняльну таблицю із рядом інших систем тестування і навчальних комплексів. Отримана таблиця показала в середньому кращі результати, а ніж інші досліджувані програмні продукти, що вказують на доцільність проведеної роботи із розробки інформаційної системи.

ВИСНОВКИ

В розглянутій магістерській дисертації було виконано дослідження способу реалізації навчально-методичного комплексу для інформаційної системи автоматизації навчального процесу, та його тестування і оцінка якості роботи. Так як розробка відповідного методу і огляд моделі отриманого комплексу потребує розгляд окремих компонентів системи, робота складалася із декількох частин.

Першим кроком до реалізації заданого навчального комплексу було огляд теоретичної складової поняття інформаційна система. Було виокремлено декілька основних частин, з яких складається інформаційна система, та зазначено відповідну різницю із іншими областями інформаційних технологій. Як складову призначення інформаційної системи визначено розробка рішень для автоматизації освітнього процесу. На основі цих даних було виокремлено ряд існуючих рішень у різних університетах світу, та побудовано порівняльну характеристику розроблених властивостей у розглянутих системах. Крім того, було зазначено які системи були розроблені на комерційній основі, а які були створені як власна розробка університету.

Другий крок дослідження передбачав пошук основних компонентів, з яких складається навчально-методична складова будь-якого вищого навчального закладу. Саме на цьому кроці було висвітлено новизну розробки, яка полягає у побудові власної структур платформи та її компонентів, із дотриманням сучасних технологій програмування і проєктування із можливістю масштабування під час росту кількості необхідних даних для аналізу, або від кількості користувачів платформи. Для вирішення цієї задачі за основу архітектури було застосовано мікросервісну складову проєкту, що передбачає розподіл компонентів за окремим функціоналом, які можуть легко взаємодіяти один з одним, і та вирішувати задачу тестування і масштабування на багато краще, а ніж інші існуючі архітектури.

На третьому кроці за отриманою моделлю платформи і навчально-методичного комплексу було виконано програмну реалізацію усіх її

компонентів. Для розробки окремих сервісів, які можуть легко взаємодіяти один із одним було вибрано систему контейнеризації Docker із застосуванням технології Docker Compose, а також зазначено відповідні конфігураційні файли проєкту. Крім цього, було показано основні сервіси, необхідні для роботи платформи, а також застосовані елементи тестування і моніторингу, які необхідні для аналізу роботи розробленого додатку. Так, усі компоненти, побудовані в рамках окремих контейнерів, які необхідні для роботи платформи і навчально-методичного комплексу є наступні:

- Основна база даних PostgreSQL для роботи додатку;
- Веб-сервер для передачі даних із платформи її користувачам nginx. А також окремо розроблена бізнес логіка програми на основі мови програмування PHP, в рамках окремого контейнеру із інтерфейсом FastCGI – PHP-FPM;
- Об’єктне сховище даних для збереження навчальних і методичних матеріалів освітнього процесу MinIO;
- Моніторингова система стану роботи компонентів Prometheus;
- Сервіс візуалізації даних перевірки стану компонентів на основі метрик Grafana;
- Сервіс перевірки якості написання коду програми SonarQube. Для його роботи було також розміщено окрему базу даних PostgreSQL, а також контейнер-клієнт, який перевіряє власне код програми Sonar Scaner;
- Ряд окремих сервісів, компонентів і клієнтських бібліотек, необхідних для збору метрик роботи усієї платформи.

Як останній крок, було проведено тестування і аналіз отриманих результатів роботи як і компонентів платформи, так і відповідність його визначеним базовим характеристикам програмного продукту. Результати, отримані із графіків окремих метрик системи показали правильну роботу як і платформи, так і її компонентів, а перевірка стану написання коду вказала, що стиль написання лежить в межах норми. Порівняння за характеристиками якості роботи програми за схожими темами роботи вказують на те, що розроблений додаток має в

середньому кращий результат за аналогічні схожі рішення, доступні на ринку продуктів.

Подальше використання платформи передбачає розробку додаткових модулів, а також покращення існуючих, які були показані у відповідності до моделі, представленої у 2 розділі. Крім того, важливою складовою правильної роботи платформи також передбачає його підтримка роботи, а також усіх її частин, з яких вона складається. Серед окремих додаткових складових можна назвати модуль Аспірантура, окремий функціонал модулю розсилки повідомлень, тощо, але аналіз і розробка таких компонентів лежить поза межами даного дослідження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Буйницька О. П., Використання електронних навчально-методичних комплексів у процесі фахової підготовки студентів [Електронний ресурс] // Інформаційні технології і засоби навчання, 2011 - №5 (25). – режим доступу: http://elibrary.kubg.edu.ua/id/eprint/641/1/O_Buynytska_ITZN_25_NDLIO.pdf
2. Буйницька О. П., Портативний електронний комплекс навчальної дисципліни + CD- диск: навч.-метод. посібник [Електронний ресурс] / Буйницька О. П., Камінський М. О. – Кам’янець-Подільский: ПП Буйницький, 2012. – 20 с. – режим доступу: http://elibrary.kubg.edu.ua/id/eprint/6253/1/BOP_NDLIO_pos2012.pdf
3. Волкова Н. П. Педагогіка : навч. посіб. [для студ. вищ. навч. закл.] / Наталія Павлівна Волкова. – [2-ге вид., перероб., доп.]. – К. : Академ. видав., 2007. – 616 с.
4. ДСТУ ISO 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. [Чинний від 2011-03]. ISO/IEC JTC 1/SC 7. Software and systems engineering, 2011. (Інформація та документація).
5. Жуков І. А., Клименко І. А., Кравець І. М. Адміністративно-навчальна інформаційна система інституту комп’ютерних технологій [Електронний ресурс]. // Проблеми інформатизації та управління. – 2007. – №19(1). – режим доступу: https://doi.org/10.18372/2073-4751.1.9059
6. Іванющенко М. С. Вибір архітектури системи управління навчальним процесом [Електронний ресурс]. : автореферат до випускної роботи. // ДонНТУ. – м. Донецьк, 2010. – режим доступу: http://masters.donntu.org/2010/fknt/ivanyushchenko/diss/indexu.htm
7. Кадемія М. Ю. Інформаційно-комунікаційні технології навчання : термінологічний словник / М. Ю. Кадемія. – Львів : СПОЛОМ, 2009. – 260 с.
8. Положення про організацію освітнього процесу в КПІ ім. Ігоря Сікорського. [Електронний ресурс]. // НТУУ КПІ ім. Ігоря Сікорського – Київ, 2020 – режим доступу: https://document.kpi.ua/files/2020_7-124.pdf

9. Ткачук В. В., Семеріков С. О., Єчкало Ю. В. Створення електронних навчально-методичних комплексів у мобільно орієнтованому середовищі навчання ВНЗ [Електронний ресурс] / В. В. Ткачук, С. О. Семеріков, Ю. В. Єчкало // Новітні комп'ютерні технології. – 2017. – 15 – ст. 189-196. – режим доступу: https://ccjournals.eu/ojs/index.php/nocote/article/view/655
10. Горбаченко И. М. Оценка качества программного обеспечения для создания систем тестирования [Електронний ресурс]. / И. М. Горбаченко. // Фундаментальные исследования. – 2013. – № 6 (4). – с. 823-827. – режим доступу: https://www.fundamental-research.ru/ru/article/view?id=31642
11. Гостев В. М., Латыпов Р. Х. Образовательная информационная среда факультета ВМК Казанского Федерального Университета: опыт формирования и развития [Електронний ресурс]. // Современные информационные технологии и IT-образование. – 2010 – с. 220-225. – режим доступу: https://cyberleninka.ru/article/n/obrazovatelnaya-informatsionnaya-sreda-fakulteta-vmk-kazanskogo-federalnogo-universiteta-opyt-formirovaniya-i-razvitiya/viewer
12. Гурьянова Т. Н., Каримова Л. К. Применение информационных систем в образовательной, научной и административной деятельности вуза (на примере КФУ) [Електронний ресурс]. // Вестник Казанского технологического университета. – 2014 – с. 381-383 – режим доступу: https://cyberleninka.ru/article/n/primenenie-informatsionnyh-sistem-v-obrazovatelnoy-nauchnoy-i-adminictrativnoy-deyatelnosti-vuza-na-primere-kfu/viewer
13. Евдокимов М. А., Мищенко А. Г. Информационная система оперативного контроля успеваемости и посещаемости студентов. Опыт разработки и внедрения [Електронний ресурс]. // Вестник Самарского государственного технического университета. Серия: Психолого-педагогические науки. – 2010. – с. 49-54. – режим доступу: https://cyberleninka.ru/article/n/informatsionnaya-sistema-operativnogo-kontrolya-uspevaemosti-i-poseschaemosti-studentov-opyt-razrabotki-i-vnedreniya/viewer

14. Информационная система «Универис» [Електронний ресурс]. // Вычислительный центр Южно-Уральского государственного университета. – режим доступу: https://www.univeris.susu.ru/Documents/%D0%9F%D1%80%D0%B5%D0%B7%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F%20%D0%9A%D0%98%D0%90%D0%A1%20%D0%A3%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D0%B8%D1%81.pptx
15. Сенькин В. В. Возможности информационных систем в управлении образованием [Електронний ресурс]. // Вестник Южно-Уральского государственного университета. Серия: Образование. Педагогические науки – 2012. – №41(18). – с. 42-45. – режим доступу: https://cyberleninka.ru/article/n/vozmozhnosti-informatsionnyh-sistem-v-upravlenii-obrazovaniem/viewer
16. Хатаева Р. Структурные компоненты автоматизированной системы управления современного инновационного вуза [Електронний ресурс]. // Вестник университета – 2015. - №5. – с. 264-268. – режим доступу: https://cyberleninka.ru/article/n/strukturnye-komponenty-avtomatizirovannoy-sistemy-upravleniya-sovremennogo-innovatsionnogo-vuza/viewer
17. Banks, E. What does “scale out” vs. “scale up” mean? [Електронний ресурс]. / Ethan Banks // Packet Pushers Interactive LLC. – 2017. – режим доступу: https://packetpushers.net/scale-up-vs-scale-out/
18. Bondi, A. B. Characteristics of scalability and their impact on performance [Електронний ресурс]. / Bondi, André B. // WOSP '00: Proceedings of the 2nd international workshop on Software and performance. – 2000. – Ottawa, Ontario, Canada – pp. 195-203. – режим доступу: https://doi.org/10.1145/350391.350432
19. Bourgeois D. T., Smith J. L., Wang S., Mortati J. What is an Information system? [Електронний ресурс]. – pp. 3-8 / Information Systems for Business and Beyond // Open Textbooks. – Saylor Academy, 2019 – p. 323. – режим доступу: https://digitalcommons.biola.edu/open-textbooks/1/

20. Brazil B. Prometheus Up & Running. / Brian Brazil. – K: Sebastobol, CA., US: O'Reilly Media Inc., 2018 – 604 с.
21. Butskyi Y. Integration of the anti-plagiarism module for information educational platforms [Електронний ресурс]. / Y. Butskyi. // Science and Technology of the XXI Century: Proceedings of the XXI International Students R&D Online Conference (Kyiv, 17 December, 2020). – Kyiv, 2020. – Part III. – pp. 20-21. – режим доступу: https://kamts2.kpi.ua/sites/default/files/Part_III_Section_9_1.pdf
22. Butskyi Y. Development prospects of universities' educational and methodological information systems on the example pf the "KPI-Connect" platform [Електронний ресурс]. / Y. P. Butsyi, K. M. Hryshchenko, I. A. Klymenko. //. I International Scientific and Theoretical Conference "Formation of Innovative Potential of World Science" (Tel Aviv, State of Israel, 7 May, 2021). – Tel Aviv, 2021. – Vol. 1. – pp. 163-170. – режим доступу: https://doi.org/10.36074/scientia-07.05.2021
23. Campbell G. A., Papapetrou P. P. SonarQube in Action. / G. Ann Campbell, Patroklos P. Paparetrou. – K.: Shelter Islands, NY., US. Manning Publications Co., 2014 – 365 с.
24. Docker Overview [Електронний ресурс]. // Docker Inc., 2021. – режим доступу: https://docs.docker.com/get-started/overview/
25. Docker Compose Overview [Електронний ресурс]. // Docker Inc., 2021. – режим доступу: https://docs.docker.com/compose/
26. Ellingwood J. Docker and Containerization [Електронний ресурс]. // The Docker Ecosystem: An Introduction to Common Components. – режим доступу: https://www.digitalocean.com/community/tutorials/the-docker-ecosystem-an-introduction-to-common-components
27. Estdale J., Georgiadou E. Applying the ISO/IEC 25010 Quality Models to Software Product [Електронний ресурс]. / X. Larrucea, I. Santamaria, R. O'Connor, R. Messnarz. // Software and Services Process Improvement. EuroSPI 2018. Communications in Computer and Information Science. Springer, Cham. – 2018. – №896. – с. 492-503. – режим доступу: https://doi.org/10.1007/978-3-319-97925-0_42

28. Gehlawat, M. School Management Information System: An Effective Tool for Augmenting the School Practices [Електронний ресурс] // New Frontiers in Education: International Journal of Education & Research. – №47. – pp. 57-64. – режим доступу: https://www.researchgate.net/publication/315380267_School_Management_Information_System_An_Effective_Tool_for_Augumenting_the_School_Practices
29. Google Trends. Порівняльна статистика популярності пошукових запитів у системі Google [Електронний ресурс]. // Google, Inc. – 2021. – режим доступу: https://trends.google.com/trends/explore?date=all&q=docker,VMware,Hyper-V,Kubernetes,KVM
30. Knorr Eric. What is cloud computing? Everything you need to know now. - [Електронний ресурс]. // – режим доступу: https://www.infoworld.com/article/2683784/what-is-cloud-computing.html
31. Lewy J, Management Information Systems in Education [Електронний ресурс] // 2020. – режим доступу: https://www.iris.co.uk/blog/management-information-systems-in-education/
32. myClarion Student Information System [Електронний ресурс]. // Clarion University – режим доступу: https://www.clarion.edu/about-clarion/computing-services/myclarion/index.html
33. myCUinfo Available Features [Електронний ресурс]. // University of Colorado Boulder – режим доступу: https://spot.colorado.edu/~mycuinfo/help/features.html
34. Prometheus Monitoring: The Definitive Guide in 2019 [Електронний ресурс]. // devconnected. – 2019. – режим доступу: https://devconnected.com/the-definitive-guide-to-prometheus-in-2019/
35. Puder Arno, Römer Kay, Pilhofer Frank. Distributed Systems Architecture. A Middleware Approach – San Francisco, Ca., US: Elsevier, 2006 – 342 p.
36. Richards, M. Software Architecture Patterns. / Michael Richards. – K: Sebastobol, CA., US: O'Reilly Media Inc., 2015 – 47 c.
37. Sastry M. Management Information Systems for Higher Education Institutions: Challenges and Opportunities [Електронний ресурс] / Sony M, Karingada K. T.,

Baporikar N. // Quality Management Implementation in Higher Education: Practices, Models, and Case Studies – Hershy, PA, US: IGI Global, 2020 – pp. 110-131 – режим доступу: https://doi.org/10.4018/978-1-5225-9829-9.ch006

38. Siraj ul Haq. Introduction to Monolithic Architecture and Microservices Architecture [Електронний ресурс]. / Siraj ul Haq // KoderLabs, Medium. – 2018. – режим доступу: https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63
39. Richards, M. Software Architecture Patterns. / Michael Richards. – K: Sebastobol, CA., US: O'Reilly Media Inc., 2015 – 47 с.
40. Types of Information System: TPS, DSS & Pyramid Diagram [Електронний ресурс] // Guru99, 2021. – режим доступу: https://www.guru99.com/mis-types-information-system.html
41. Wallace, P. Information System in Action. – pp. 4-9 // Introduction to Information Systems. Third Edition. – New Jersey, US: Pearson Education, Inc., 2018 – p. 420.
42. Zwass, V. Information system [Електронний ресурс]. // Encyclopedia Britannica. – режим доступу: https://www.britannica.com/topic/information-system